Visuelle Qualitätsmetrik basierend auf der multivariaten Datenanalyse von H.264/AVC Bitstream-Features

Technical Report

Manuel Klimpke, Christian Keimel and Klaus Diepold

Visuelle Qualitätsmetrik basierend auf der multivariaten Datenanalyse von H.264/AVC Bitstream-Features

Technical Report

Manuel Klimpke, Christian Keimel and Klaus Diepold

25. November 2010

Lehrstuhl für Datenverarbeitung
Technische Universität München

Manuel Klimpke, Christian Keimel and Klaus Diepold. *Visuelle Qualitätsmetrik basierend auf der multivariaten Datenanalyse von H.264/AVC Bitstream-Features. Technical Report.* Technische Universität München, München, 2012.

Lehrstuhl für Datenverarbeitung, Technische Universität München, 80290 München, http://www.ldv.ei.tum.de.

Zusammenfassung

In dieser Diplomarbeit wird eine visuelle *'no-reference'* Qualitätsmetrik für H.264/AVC HD-Videomaterial vorgestellt, die ausschließlich über Informationen aus dem Bitstream eine Schätzung der Videoqualität vornimmt. Dazu wurde auf Basis der H.264/AVC-Referenzsoftware ein *'feature extractor'* implementiert, der aus dem H.264-Bitstream 64 charakteristische Merkmale ('Bitstream-Features') ausliest. Dies waren unter anderem die Bitrate, die Quantisierungsparameter, das Verhältnis der verschiedenen Slice- und Makroblock-Typen sowie diverse Eigenschaften der Bewegungsvektoren.

Die Modellierung der Qualitätsmetrik erfolgte mit der Methode der multivariaten Datenanalyse, welche anhand einer 'Partial Least Squares Regression' aus den Trainingsdaten von insgesamt 32 Videosequenzen ein Regressionsmodell erstellte, das anschließend über das Kreuzvalidierungsverfahren überprüft und optimiert wurde. Das optimierte Modell, über das die Qualität einer unbekannten Videosequenz vorausgesagt werden kann, basierte auf 48 unterschiedlich stark gewichteten Bitstream-Features.

Abstract

This thesis presents a visual no-reference quality metric for H.264/AVC high definition video, which estimates the video quality exclusively with information directly extracted from the bitstream. Based on the H.264/AVC reference software a 'feature-extractor' was implemented to extract 64 characteristic bitstream-features (e.g. bitrate, quantization parameters, ratio of different slice- and macroblock-types and different attributes of the motion vectors).

The quality metric was built with multivariate data analysis: A partial least squares regression model was calibrated with training data from 32 video sequences and subsequently validated and optimized through cross-validation. The final model contains 48 variously weighted bitstream-features to predict the quality of unknown video.

Inhaltsverzeichnis

1 Motivation

„HD Videoaufnahmen. Das Leben sieht besser aus in HD.[1]“

Mit diesem Spruch wirbt ein bekanntes Unternehmen für sein aktuelles Smartphone, das laut Hersteller Videos *„in beeindruckender HD-Qualität“* aufzeichnen kann. Bei Versprechen wie diesen sollte man sich die Frage stellen, inwieweit die visuelle Qualität dieser Videodateien außer der Grundvoraussetzung (HD-Auflösung 1280×720) wirklich dem „High Definition“-Standard genügt.

Zwar ist es dem Video-Codec H.264/AVC gelungen, einen neuen Meilenstein in effizienter Videokompression zu setzen, aber natürlich stößt auch dieser Standard an Grenzen - denn einerseits muss ein verlustbehafteter Videokodierungsalgorithmus immer einen Kompromiss zwischen hoher Bildqualität und niedriger Datenmenge eingehen (wobei die Tendenz gerade bei Smartphones aufgrund der begrenzten Speicherkapazität zu letzterem Punkt neigt), und andererseits kann die visuelle Qualität natürlich nicht durch die Kodierung verbessert werden, falls das Ausgangsmaterial des Bildsensors schon keine gute Qualität liefert. Außerdem können gerade bei Smartphones sehr rechenintensive Profile und Optionen von H.264/AVC (siehe Kapitel 2.3) aufgrund der begrenzten Systemleistung oftmals nicht genutzt werden, wodurch die Kodierung nicht die optimalen Ergebnisse erzielen kann.

Doch wie bewertet man eigentlich die Qualität von Videodateien?

Wie in Abschnitt 2.1 noch genauer erläutert wird, muss für eine möglichst objektive und neutrale Beurteilung der Bildqualität einer Videosequenz meist auf die subjektiven Bewertungen durch mehrere Versuchspersonen zurückgegriffen werden, was jedoch einen nicht unerheblichen Aufwand erfordert und sowohl Zeit als auch Kosten verschlingt.

Die vorliegende Diplomarbeit setzt an diesem Punkt an und versucht, die subjektive Qualitätswahrnehmung des Menschen zu erlernen und über eine visuelle Qualitätsmetrik nachzubilden, mit der eine Qualitätsschätzung von Videosequenzen über einen automatisierten Algorithmus ermöglicht wird. Desweiteren war die Vorgabe, dass die Qualitätsevaluierung ausschließlich auf Informationen aufbaut, die direkt aus dem Bitstream des H.264-kodierten Videomaterials gewonnen wurden. Somit sollte die rechenaufwändige Dekodierung und Rekonstruktion der Einzelbilder für die Qualitätsbeurteilung nicht nötig sein, was eine leistungsfähige und ressourcenschonende Implementierung erlaubt.

Um diese Vorgaben umzusetzen, kann die Arbeit in zwei Teilprojekte gegliedert werden: Die Merkmalsextraktion und die multivariate Datenanalyse.

[1] Quelle: `http://www.apple.com/de/iphone/` am 16.11.2010.

Ziel des ersten Teilprojektes war es, einen '*feature extractor*' zu implementieren, der aus dem Bitstream einer H.264-kodierten HD-Videosequenz möglichst viele charakteristische Merkmale ('Bitstream-Features') ausliest. Anschließend sollte über die Bitstream-Features von 32 Testsequenzen, zusammen mit den Qualitätsbeurteilungen dieser Testsequenzen, welche in subjektiven Videotests am Lehrstuhl für Datenverarbeitung an der Technischen Universität München ermittelt wurden, durch eine multivariate Datenanalyse ein Modell gefunden werden, das die Bildqualität von unbekannten Videosequenzen über die extrahierten Bitstream-Features möglichst genau abschätzt.

Abbildung 1.1 gibt einen Überblick über den Ansatz und die soeben beschriebene Vorgehensweise dieser Arbeit.

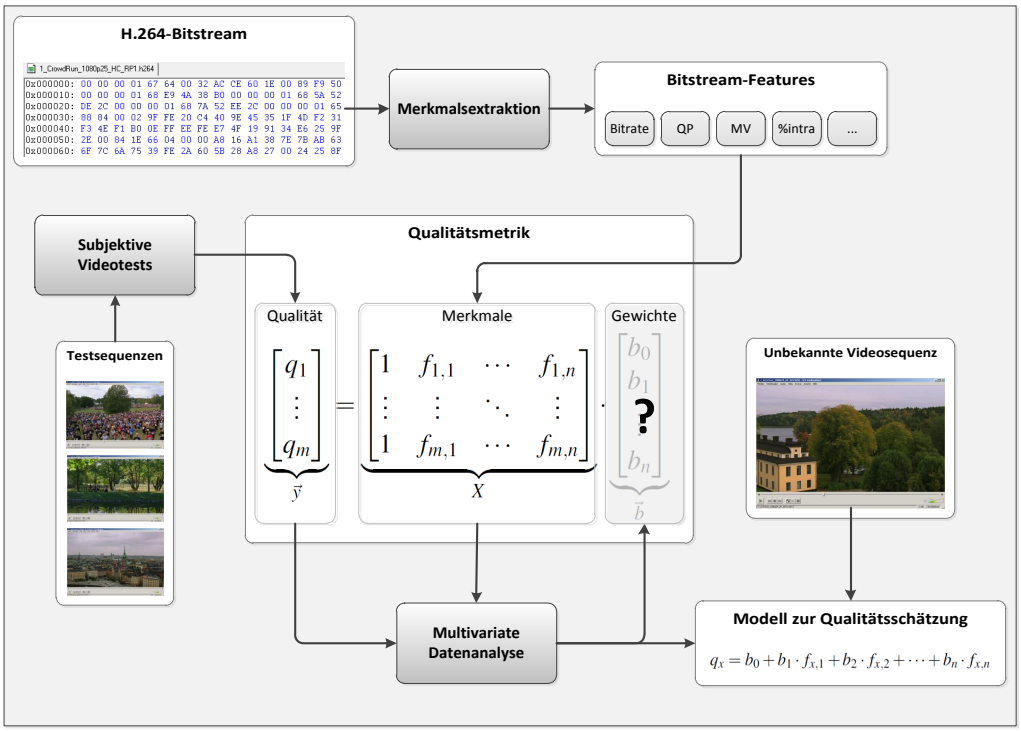


Abbildung 1.1: Gesamtübersicht über die Aufgabenstellung der Diplomarbeit

Nach einer Einführung in die zum Verständnis benötigten Grundlagen der Videokodierung im nachfolgenden Kapitel 2 widmet sich Abschnitt 3 dem ersten Meilenstein dieser Arbeit, nämlich der Merkmalsextraktion aus dem H.264-Bitstream. Anschließend wird unter Kapitel 4 mithilfe der multivariaten Datenanalyse die Qualitätsmetrik modelliert, deren Ergebnisse in Abschnitt 5 präsentiert und evaluiert werden. Abschließend wird eine kurze Zusammenfassung sowie ein Ausblick auf Verbesserungsmöglichkeiten gegeben.

2 Grundlagen der Videokodierung nach dem H.264/AVC-Standard

Im folgenden Abschnitt 2.1 werden die verschiedenen Methoden der Qualitätsevaluierung von Videosequenzen vorgestellt, bevor in Kapitel 2.2 die Grundlagen der hybriden Videokodierung kurz eingeführt werden. Anschließend wird unter 2.3 die Funktionsweise des Videokodierungs-Standard H.264/AVC erklärt und dessen Bitstream-Syntax in Abschnitt 2.4 erläutert.

2.1 Methoden der Qualitätsevaluierung von kodierten Videosequenzen

Die visuelle Wahrnehmung von Bildfolgen ist äußerst komplex und kann bis heute nicht durch ein mathematisches Modell exakt nachgebildet werden. Gerade bei sich rasch ändernden Szenen, Sequenzen mit viel Bewegung oder detail- und kontrastreichen Elementen spielen Maskierungseffekte eine große Rolle, wodurch das menschliche Auge keinen linearen Gesetzmäßigkeiten mehr gehorcht. Aus diesem Grund ist die Bewertung der Bildqualität von komprimierten Videosequenzen alles andere als ein triviales Problem.

Da letztendlich der subjektive Qualitätseindruck einer Videodatei auf den Menschen entscheidend ist, bedient man sich für die Bewertung von komprimiertem Videomaterial falls möglich der subjektiven Qualitätsevaluierung: Dabei bewerten Versuchspersonen eine Reihe von verschiedenen Videosequenzen unterschiedlicher Qualität auf einer standardisierten Skala. Damit derartige Videotests reproduzierbare und vergleichbare Ergebnisse liefern, sind die Rahmenbedingungen (z.B. Mindestanzahl an Versuchspersonen, Raumbeleuchtung, Displayparameter etc.) von der *Video Quality Experts Group* (VQEG) genau festgelegt [1].

Der große Nachteil von subjektiven Qualitätsevaluierungen liegt jedoch auf der Hand: Durch den Einsatz von Versuchspersonen sind sie teuer, zeitaufwändig und für den Echtzeitbetrieb in der Praxis meist nicht geeignet.

Die Wissenschaft versucht daher seit geraumer Zeit, die Bewertungskriterien des Menschen möglichst genau auf einen Algorithmus zu adaptieren und dadurch eine sogenannte „objektive Qualitätsevaluierung“ von kodiertem Videomaterial zu ermöglichen. Bei derartigen Ansätzen unterscheidet man zwischen drei verschiedenenen Verfahren: *'Full-reference'* (FR), *'reduced-reference'* (RR) und *'no-reference'* (NR).

Bei Ersterem wird die gesamte komprimierte Videosequenz mit der gesamten ungestörten Referenzsequenz verglichen, beim RR-Verfahren werden nur einige Merkmale aus beiden Sequenzen extrahiert und verglichen.

Da jedoch oftmals die unkomprimierte Originalsequenz nicht verfügbar ist, sondern lediglich das gestörte Video auf der Empfangseite, muss in diesen Fällen auf das NR-Verfahren zurückgegriffen werden. Ein weitverbreiteter Ansatz versucht dabei, aus der dekodierten Videosequenz charakteristische Merkmale zu extrahieren und durch diese Rückschlüsse auf die Bildqualität zu ziehen. Ein wichtiges Werkzeug, das in Verbindung mit objektiven Qualitätsevaluierungen oft verwendet wird, ist das *Peak signal to noise ratio* (PSNR), welches im nachfolgenden Abschnitt genauer vorgestellt wird.

Berechnung des PSNR

Das *Peak signal to noise ratio* (PSNR - auf deutsch etwa „Spitzen-Rauschabstand“) beschreibt das Verhältnis zwischen Maximalamplitude eines Referenzsignals und der tatsächlichen Amplitude des komprimierten Signals und kann damit als Maß für die Bildqualität von kodiertem Videomaterial verwendet werden.

Das PSNR berechnet sich folgendermaßen:

$$\text{PSNR} = 10 \cdot log_{10}(\frac{I_{max}^2}{\text{MSE}})dB \tag{2.1}$$

Dabei ist I_{max} die maximale Intensität, was bei einem mit 8 Bit quantisierten Bild dem maximalen Pixelwert 255 entspricht. MSE steht für die mittlere quadratische Abweichung (*mean squared error*) und kann nach Gleichung 2.2 berechnet werden:

$$\text{MSE} = \frac{1}{XY}\sum_{x=0}^{X-1}\sum_{y=0}^{Y-1}(p_{Ref}(x,y) - p_{Komp}(x,y))^2 \tag{2.2}$$

X entspricht der horizontalen, Y der vertikalen Auflösung des (monochromen) Bildes, $p_{Ref}(x,y)$ und $p_{Komp}(x,y)$ sind die Pixelwerte des unkomprimierten Referenzbildes bzw. des durch Komprimierung gestörten Bildes an der Stelle x,y. Bei Farbbildern mit drei Werten pro Pixel wird der MSE für jeden Farbkanal einzeln berechnet und anschließend der Mittelwert gebildet.

Obwohl das PSNR nicht die nichtlinearen Eigenschaften und Eigenheiten der menschlichen Wahrnehmung von Bildfolgen berücksichtigt, ist es dennoch ein einfaches und brauchbares Werkzeug, um die Qualität von kodiertem Videomaterial abschätzen zu können [7]. Vereinfacht gesagt gilt: Je höher das PSNR, desto besser ist die Videoqualität. Typische Werte des PSNR von qualitativ hochwertigen Videodateien liegen zwischen 30 und 40 dB.

2.2 Grundlagen, Werkzeuge und Begriffe der hybriden Videokodierung

Eine unkomprimierte digitale HD-Videosequenz mit der Auflösung 1920 $\times$ 1080 und 25 Einzelbildern[1] pro Sekunde (*'frames per second'* - fps) würde im RGB-Farbraum bei 10 Bit Quantisierung zur Speicherung oder Übertragung ca. 2,2 Gbit/s in Anspruch nehmen. Da dies selbst im professionellen Bereich oftmals zu viele Ressourcen benötigt, wird als erster Kompressionsschritt meist die Farbinformation unterabgetastet.

2.2.1 Farbunterabtastung

Seit es das Farbfernsehen gibt, erfasst eine Farbvideokamera für jeden Bildpunkt (*pixel*) die drei Farbkomponenten Rot, Grün und Blau. Diese Bildinformationen werden normalerweise direkt nach der Aufzeichnung vom RGB-Farbraum in den YC_bC_r-Farbraum konvertiert, was anfangs den einfachen Grund hatte, das Signal auch für Schwarz-Weiss-Fernseher kompatibel zu machen.

Y enthält dabei die Helligkeitsinformation (Luminanz) und kann aus den drei Farbwerten R, G und B über die Gleichung 2.3 für HD-Video [2] berechnet werden [14]:

$$Y = 0,2126 \cdot R + 0,7152 \cdot G + 0,0722 \cdot B \tag{2.3}$$

Die unterschiedliche Gewichtung der Farbwerte resultiert aus der spektral unterschiedlichen Helligkeitsempfindung des menschlichen Auges, welche bei einer Wellenlänge von 555 nm (entspricht grün) ein signifikantes Maximum aufweist [6]. Die beiden Farbdifferenzwerte C_b und C_r stehen für die Farbkomponenten (Chrominanz) Blau und Rot und drücken vereinfacht gesagt die Differenz $B - Y$ respektive $R - Y$ aus.

In der Praxis werden Y, C_b und C_r über eine 3 $\times$ 3-Matrix-Multiplikation aus R, G und B berechnet und auf der Empfangsseite R, G und B aus dem YC_bC_r-Signal entsprechend zurückberechnet.

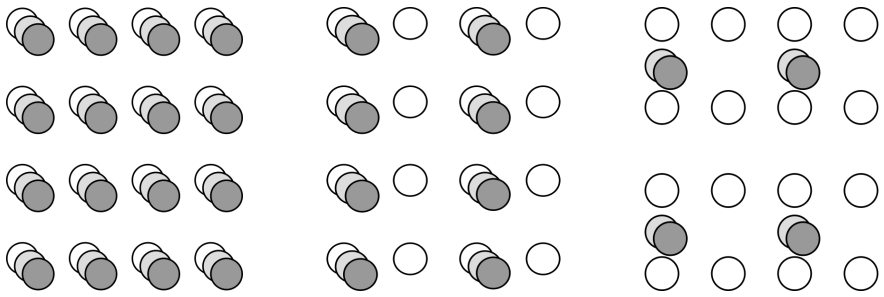
Da das menschliche Auge deutlich empfindlicher gegenüber Luminanz als Chrominanz ist, kann bei der Verwendung des YC_bC_r-Farbraums die Farbinformation unterabgetastet werden:

Anstatt für jeden Pixel alle drei Komponenten Y, C_b und C_r zu übertragen (sogenannte 4:4:4-Abtastung, siehe Abb. 2.1a), werden bei der 4:2:2-Abtastung die Chrominanz-Werte C_b bzw. C_r nur für jeden zweiten Bildpunkt in horizontaler Richtung übertragen (siehe Abb. 2.1b). Bei der 4:2:0-Abtastung wird zusätzlich auch in vertikaler Richtung nur jeweils ein C_b- bzw. C_r-Wert für zwei Pixel berechnet (siehe Abb. 2.1c).

Obwohl dies ein verlustbehafteter Vorgang der Datenreduktion ist, hat er aufgrund des angesprochenen Sehempfindens des Menschen kaum Auswirkungen auf den subjektiven

[1] Da im Rahmen dieser Arbeit ausschließlich progressives Videomaterial verwendet wurde, sind nachfolgend immer Vollbilder gemeint, wenn von Einzelbildern (*frames*) gesprochen wird.

[2] Für SD-Video (PAL) gelten andere Gewichtungen: $Y = 0,299 \cdot R + 0,587 \cdot G + 0,114 \cdot B$



(a) 4:4:4-Abtastung **(b)** 4:2:2-Abtastung **(c)** 4:2:0-Abtastung

Abbildung 2.1: Verschiedene Arten der Farbabtastung. Von links nach rechts nimmt der Datenumfang und damit die visuelle Qualität ab, während die Effizienz der Datenkompression steigt. (Bildquelle: [24])

Qualitätseindruck der Videosequenz, verringert gleichzeitig jedoch die Datenrate signifikant (um den Faktor 0,33 bei 4:2:2-Abtastung bzw. 0,5 bei 4:2:0-Abtastung).

Aus diesem Grund wird auch im professionellen Bereich überwiegend mit 4:2:2-Material gearbeitet, was die Datenrate im zu Beginn des Kapitels 2.2 erwähnten Beispiel (HD-Videosequenz, Auflösung 1920x1080, 25 fps) auf 1,485 Gbit/s begrenzt und das Videomaterial damit über HD-SDI[3] übertragungsfähig macht [14].

Da diese Datenrate aber immer noch um ein Vielfaches zu hoch für die Übertragung über z.B. Internet oder Satellit ist, muss die Datenmenge weiter reduziert werden. Verschiedene Methoden zur verlustfreien oder verlustbehafteten Datenreduktion werden im folgenden Abschnitt vorgestellt.

2.2.2 Elemente von Videokodierungsalgorithmen

Allgemein gesagt, ist das Ziel von Datenreduktionsverfahren, jede redundante und überflüssige Information in einem Datenstrom zu finden und zu eliminieren. Dies kann zum einen vollständig reversibel und damit ohne Qualitätsverlust geschehen, z.B. durch geschicktes Umsortieren und Kodieren des Datenstroms, zum anderen aber auch mit dem Verlust von Informationen auf der Empfängerseite verbunden sein. Dieser Informationsverlust beeinflusst die subjektive Qualitätswahrnehmung im Idealfall allerdings so gering, dass dies billigend in Kauf genommen wird, um gleichzeitig die Datenmenge signifikant verringern zu können.

Da fast alle gängigen Videokodierungsverfahren eine Kombination verschiedener Grundbausteine sind, wird nun eine Übersicht dieser Grundelemente gegeben und deren Funktionsweise knapp erläutert.

[3] HD-SDI, ausgeschrieben „High Definition Serial Digital Interface“, ist die Standard-Schnittstelle zur Übertragung von HD-Video im professionellen Umfeld; siehe auch SMPTE 292M.

Entropiekodierung

Die Entropiekodierung stellt ein verlustloses Verfahren dar und ist normalerweise einer der letzten Schritte eines Videokodierungsalgorithmus. Das sogenannte *Variable Length Coding* (VLC) sortiert die Elemente eines Datenstroms nach Häufigkeit und ersetzt diese Elemente durch Abkürzungen, die möglichst kurz sind, je häufiger das Element auftritt.

Wird das Abkürzungs-Alphabet zusammen mit dem kodierten Datenstrom übertragen, kann der Empfänger eine Rückübersetzung durchführen. Das bekannteste VLC-Verfahren ist die Huffman-Kodierung.

Ein weiterer Teil der Entropiekodierung ist das *Run Length Encoding* (RLE). Sortiert man vorab den Datenstrom über den sogenannten Zick-Zack-Scan geschickt um, so dass viele gleiche Elemente aufeinanderfolgen, ist es ausreichend, jedes Element nur einmal - mit dem Hinweis, wie oft es nacheinander auftritt - zu kodieren, anstatt alle sich wiederholenden Elemente separat zu übertragen. Ein Beispiel: Die Bitfolge $'1111000000'$ kann durch $'4 \times 1; 6 \times 0'$ um einiges effizienter beschrieben werden.

Intrakodierung

Betrachtet man ein Einzelbild (*frame*) einer Videodatei, fällt häufig auf, dass benachbarte Bildpunkte eine große Ähnlichkeit aufweisen. Abgesehen von einigen wenigen Kanten findet man im Bild oftmals grobe Strukturen mit annähernd gleichen Farb- und Helligkeitswerten und sanften Übergängen zwischen den einzelnen Bildpunkten (z.B. Himmel/Hintergrund).

Diese örtliche Korrelation nutzt die Intrakodierung zur Datenreduktion, indem jeder Bildpunkt und dessen Luminanz-/Chrominanzinformation nicht isoliert abgespeichert wird, sondern das Einzelbild in (meist quadratische) Blöcke der Größe $M \times N$ unterteilt wird; als Kompromiss zwischen Rechenaufwand und Kompression hat sich dabei eine Blockgröße zwischen 4×4 und 16×16 etabliert.

Anschließend gibt es mehrere Verfahren, um die Korrelation der Pixelwerte innerhalb der als Makroblöcke bezeichneten Bildabschnitte effizient auszunutzen:

Bei der 'differenziellen Puls Code Modulation' (DPCM) werden innerhalb der Blöcke nur die Differenzwerte zum Nachbarpixel kodiert, was aufgrund der angesprochenen Eigenschaften in einer deutlich kompakteren Beschreibung des Bildes resultiert.

Ein anderes, oft verwendetes Verfahren ist die 'diskrete Kosinustransformation' (DCT), welche die Werte eines Makroblocks in den Frequenzbereich transformiert.

Sowohl bei der DPCM als auch bei der DCT treten erst durch die nachfolgende Quantisierung Verluste auf. Wie stark die Daten bei der Quantisierung gerundet werden, legt eine Quantisierungsmatrix fest. Diese zumeist variablen Quantisierungsparameter sind daher die Steuerungselemente der Kompression - sie bestimmen auf der einen Seite die auftretenden Verluste, auf der anderen Seite die dadurch erzielte Datenreduktion.

Ein prominentes Beispiel für Intrakodierung ist JPEG [3]. Ein Bild wird dabei in Makro-

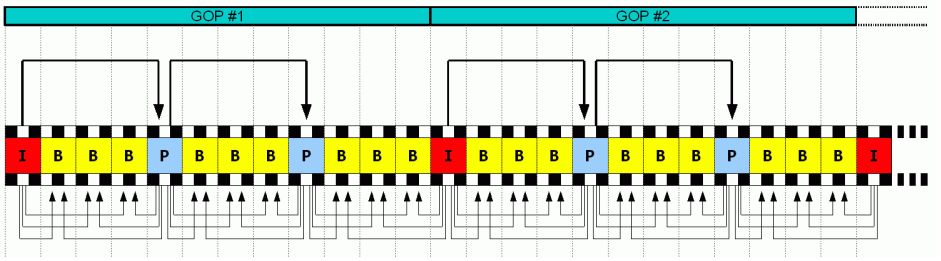


Abbildung 2.2: Beispiel einer *Group of Pictures* (GOP) der Länge 12. Die Pfeile verdeutlichen die Referenzierungen der Einzelbilder. (Bildquelle: [23])

blöcke der Grösse 8×8 unterteilt, diese Blöcke werden mithilfe der DCT transformiert und anschließend quantisiert; abschließend erfolgt eine Entropiekodierung.

Wie der Name Intrakodierung andeutet, beschränkt sich die Kodierung nur auf ein Einzelbild, was dieses sogenannte 'I-Frame' unabhängig von anderen Einzelbildern der Videosequenz macht.

Interkodierung

Im Gegensatz zur Intrakodierung nutzt die Interkodierung nicht die örtliche, sondern die zeitliche Korrelation in einer Videosequenz. Bei aufeinanderfolgenden Einzelbildern ist die Ähnlichkeit oft sehr groß; in der Regel sind diese während einer Szene bis auf wenige (bewegte) Objekte im Bild identisch.

Aus diesem Grund macht es Sinn, nur die Informationen über Änderungen und Bewegungen im Bild in Bezug auf ein benachbartes Referenz-Einzelbild zu kodieren. Dazu werden die Einzelbilder analog zur Intrakodierung in Makroblöcke unterteilt und in benachbarten Bildern die Blöcke mit der größten Ähnlichkeit gesucht. Die relative Verschiebung eines Blockes zwischen den Einzelbildern beschreibt ein Bewegungsvektor (*motion vector* - MV), dessen Übertragung zusammen mit dem Referenzblock genügt, um den Block an seiner neuen Stelle im aktuellen Einzelbild rekonstruieren zu können.

Dieses Verfahren wird auch als Schätzung der Bewegungskompensation (*motion-compensated prediction* - MCP) bezeichnet [9].

Im Zusammenhang mit der Interkodierung unterteilt man die Videosequenz in *Groups of Pictures* (GOP). Jede GOP beginnt mit einem intrakodierten I-Frame, welches als Referenzframe für die nachfolgenden Einzelbilder dient. Auf dieses I-Frame folgt entweder ein P-Frame, welches lediglich die Differenzinformationen zum vorangegangenen I-Frame oder P-Frame enthält, oder ein B-Frame, welches die Differenzinformationen zum vorausgegangenen und/oder nachfolgenden I-Frame oder P-Frame enthält.

Abbildung 2.2 zeigt ein Beispiel einer *Group of Pictures*.

Zusammenfassung

Zusammenfassend arbeitet ein hybrider Videokodierungsalgorithmus typischerweise nach folgender Prozedur [20]:

Jedes Einzelbild wird in Blöcke unterteilt. Das erste Frame einer Szene wird intrakodiert, die nachfolgenden Einzelbilder der GOP durch Bewegungsschätzung interkodiert. Die Bewegungsvektoren (MV) enthalten dabei die Informationen, wohin der jeweilige Block im nächsten Frame gewandert ist.

Die übriggebliebenen Informationen werden durch eine Frequenztransformation umgeformt und anschließend skaliert und quantisiert. Nach einer Entropiekodierung werden die am Ende stark komprimierten Daten zusammen mit den nötigen Metadaten über Art und Umfang der Kodierung übertragen.

Nachdem nun die grundlegenden Werkzeuge der hybriden Videokodierung vorgestellt wurden, kann im folgenden Kapitel die genaue Funktionsweise des hybriden Video-Codecs H.264/AVC erläutert werden.

2.3 Prinzip und Funktionen des Video-Codecs H.264/AVC

In diesem Abschnitt wird der Video-Codec H.264/AVC und seine Arbeitsweise vorgestellt. Wie in Abbildung 2.3 skizziert wurde, bezeichnet man als Codec den Teil innerhalb der Verarbeitungskette einer Videodatei von der Quelle zum Empfänger, der die Videodaten für die Übertragung kodiert (*en**co**ding*) und aus dem kodierten Datenstrom (*bitstream*) auf der Empfangsseite die Videodatei wiederherstellt (***dec**oding*).

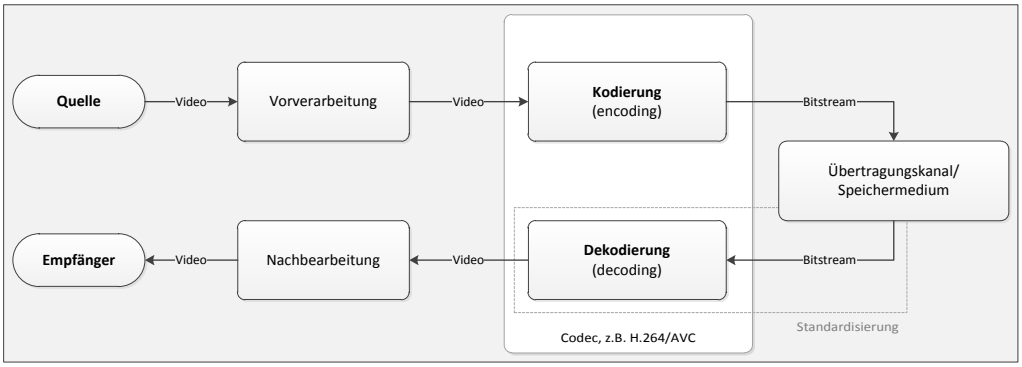


Abbildung 2.3: Verarbeitungskette einer Videodatei von der Quelle zum Empfänger. Der umrahmte Abschnitt zwischen *en**co**ding* und ***dec**oding* wird als Codec bezeichnet.

2.3.1 Übersicht über den Standard H.264/AVC

Der Standard H.264/AVC gilt wie sämtliche ITU-T und ISO/IEC JTC1 Videokodierungs-Standards nur für die Bitstream-Syntax und den Dekodierungsvorgang [12]. Dies schafft einerseits Freiräume für den Kodierungsvorgang, stellt andererseits jedoch sicher, dass jeder *decoder* das gleiche Ergebnis für eine kodierte Videosequenz liefert.

Das Kürzel AVC steht für 'Advanced Video Coding' und repräsentiert Part 10 des ISO/IEC-Standards 14496 [15], welcher besser bekannt ist als MPEG-4. Die 'Moving Pictures Expert Group' (MPEG) entwickelte in Zusammenarbeit mit der 'ITU-T Video Coding Expert Group' den Standard H.264/AVC als Nachfolger des etablierten Videokodierungs-Standard MPEG-2, welcher ISO/IEC intern mit H.262 bezeichnet wird.

Im Vergleich zu MPEG-2 soll mit H.264/AVC bei vergleichbarer Bildqualität eine zusätzliche Datenreduktion von ca. 50 Prozent erreichbar sein [22], was jedoch mit einer erhöhten Komplexität und einem größeren Rechenaufwand verbunden ist. Dies wird wie schon unter 2.2.2 angedeutet durch ein intelligentes Kombinieren von zahlreichen Bausteinen erreicht. Da viele davon optional sind und je nach Anwendungsfall aktiviert oder deaktiviert werden können, wird zudem eine hohe Flexibilität erreicht, was den Codec H.264/AVC an eine Vielzahl an unterschiedlichen Anwendungen und Übertragungskanäle anpassbar macht. Wie auch schon in vielen vorangegangenen Standards erfolgt daher eine Unterteilung in Profile und Level: Je nach Profil (Main, High, Extended, Baseline, ...) sind beim *encoding* unterschiedliche Funktionen im Algorithmus aktiviert bzw. optional zuschaltbar, die verschiedenen Level erlauben eine Abstufung hinsichtlich der angestrebten Bitrate und Auflösung der kodierten Videosequenz.

Die wesentlichen Implementierungen und Funktionen des Codec H.264/AVC werden nun im Folgenden vorgestellt; insbesondere werden dabei die Verbesserungen gegenüber Vorgängern hervorgehoben.

2.3.2 Elemente von H.264/AVC

Unterteilung der Einzelbilder in 'Slices'

Anstatt die einzelnen Bilder der Videosequenz direkt in Makroblöcke zu unterteilen, ist es bei H.264/AVC möglich, das Einzelbild zunächst in sogenannte 'Slices' zu unterteilen. Dadurch lassen sich in jedem Frame variabel Regionen oder Objekte gruppieren, die anschließend unabhängig betrachtet und unterschiedlich kodiert werden können. Im Gegensatz zu MPEG-2 kann dabei jedes Frame theoretisch in beliebig viele Slices beliebiger Form und Größe zerlegt werden, jedoch andererseits auch nur aus einem einzigen Slice bestehen. Letzteres ist in der Praxis zumeist der Fall, d.h. ein Slice entspricht in der Regel einem Frame. Statt einer Unterteilung in I-Frames, P-Frames und B-Frames kategorisiert man im Zusammenhang mit H.264/AVC meist in I-Slices, P-Slices und B-Slices.

Da jedes Slice unabhängig von den restlichen Slices des zugehörigen Einzelbildes dekodiert werden kann, kann deren Reihenfolge im Bitstream auch völlig beliebig gewählt

werden. Dieses als 'Arbitrary Slice Ordering' (ASO) bezeichnete Merkmal kommt insbesondere Echtzeitübertragungen über Internet zu Gute.

Makroblock-Partitionierung

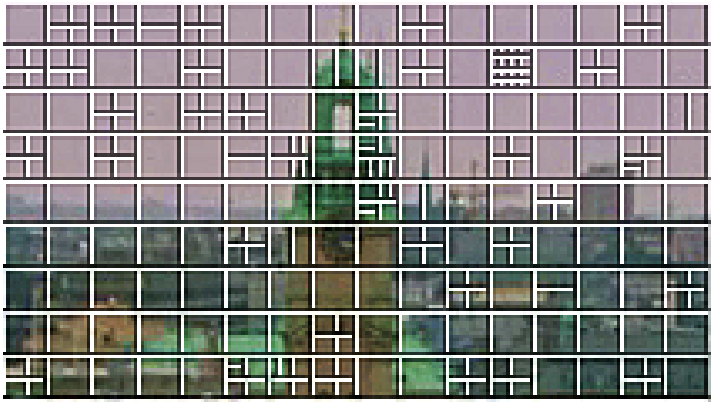


Abbildung 2.4: Makroblock-Partitionierung bei H.264/AVC am Beispiel eines B-Frame-Ausschnittes. Neben der Grundform 16×16 sind sämtliche möglichen Variationen der Blockgrößen zu erkennen.

Die Slices werden bei H.264/AVC wie gewohnt sowohl bei intrakodierten I-Slices als auch bei interkodierten P-Slices und B-Slices weiter in Makroblöcke unterteilt. Diese haben die Ausgangsgröße 16×16 Pixel, können jedoch in Unterblöcke bis zu einer Größe von 4×4 Pixeln unterteilt werden und bei interkodierten Slices für die Bewegungsschätzung auch nicht quadratische Formen annehmen (z. B. 8×16 oder 8×4, siehe Abbildung 2.4). Dies macht die Kodierung um einiges flexibler, effizienter und gleichzeitig qualitativ hochwertiger, da die Blockgröße und -form somit an die Eigenschaften des entsprechenden Bildabschnittes angepasst werden kann. In detailreichen Regionen gehen durch kleinere Blöcke weniger Informationen verloren, in groben Bereichen erzielt man durch die Wahl größerer Blöcke eine höhere Kodiereffizienz (siehe Abbildung 2.5 auf der nächsten Seite).

Analog zum 'Arbitrary Slice Ordering' unterstützt H.264/AVC das 'Flexible Macroblock Ordering' (FMO): Die Makroblöcke innerhalb eines Slices können in beliebiger Reihenfolge im Bitstream kodiert werden.

Intrakodierung mit flexibler Blockgröße

Wie im vorangegangenen Abschnitt bereits erwähnt, werden I-Slices in 16×16 Pixel große Makroblöcke aufgeteilt, welche für detailreiche Bildabschnitte nochmal in bis zu 16 Blöcke der Größe 4×4 aufgespalten werden können (siehe Abbildung 2.5 auf der nächsten Seite).

Im Gegensatz zu z. B. JPEG, wo die Intrakodierung über die DCT eine Transformation in den Frequenzbereich vornimmt, basiert die Intrakodierung bei H.264/AVC auf der Metho-

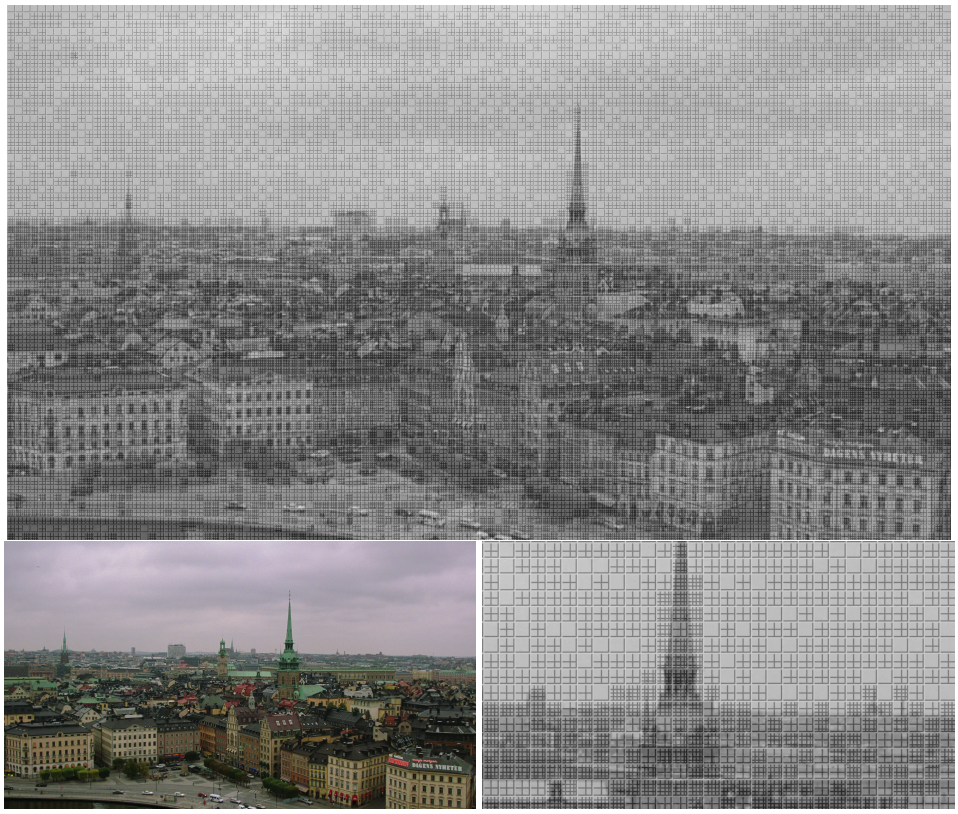


Abbildung 2.5: Makroblock-Raster eines H.264-kodierten I-Frames. In detailarmen Bildbereichen (z. B. Himmel) sind noch viele 16 $\times$ 16-Makroblöcke zu finden, während diese in komplexeren Bildabschnitten fast ausschließlich in 8 $\times$ 8 bzw. 4 $\times$ 4 Unterblöcke partitioniert werden. Unten links ist das Originalframe zu sehen, unten rechts ein vergrößerter Bildausschnitt.

de der DPCM. Dabei wird der Luminanz- bzw. Chrominanzwert des aktuellen Makroblocks über die Werte der benachbarten Blöcke links bzw. über dem aktuellen Block geschätzt und dann lediglich der Prädiktionsfehler, also die Differenz zwischen tatsächlichem Wert und geschätzten Wert, kodiert und übertragen. Um die Unabhängigkeit der einzelnen Slices zu gewährleisten, werden bei der Prädiktion jedoch keine Nachbarblöcke aus anderen Slices berücksichtigt.

Der Standard H.264/AVC beinhaltet zudem noch eine Sonderform für intrakodierte Makroblöcke: Für sogenannte I_PCM-Blöcke wird keine Schätzung vorgenommen, die Werte werden hierbei direkt kodiert. Dies ermöglicht für ausgewählte Blöcke eine verlustfreie Kodierung.

Abbildung 2.6: Interkodierung bei H.264/AVC am Beispiel zweier P-Frames. Die roten Pfeile im unteren Einzelbild sind die Bewegungsvektoren (MV, die auf die Stelle zeigen, an der sich der jeweilige Makroblock im Referenzframe (oberes Bild) befand.

Interkodierung mit präziser und gewichteter Bewegungsschätzung

Im Unterschied zur Interkodierung in früheren Standards ist bei H.264/AVC die Blockgröße für die Schätzung der Bewegungskompensation sehr flexibel:

Interkodierte Makroblöcke in P-Slices und B-Slices können wahlweise 16×16, 16×8, 8×16 oder 8×8 Pixel umfassen, 8×8-Blöcke können optional in Unterblöcke der Größe 8×4, 4×8 oder 4×4 zerlegt werden. Außerdem ist es bei H.264/AVC auch in P-Slices und B-Slices möglich, auf einzelne Makroblöcke die in I-Slices übliche Intrakodierung anzuwenden, falls dies zu einer effizienteren Kodierung führt. Damit ist eine sehr genaue Anpassung an die Gegebenheiten im Bild möglich.

Zudem wurde bei H.264/AVC die Referenzierung bei der Bewegungsschätzung deutlich variabler gestaltet:

Für die Suche des ähnlichsten Bildausschnittes kann nicht nur das unmittelbar vorangegangene (bzw. bei B-Frames auch das unmittelbar nachfolgende) I-Frame oder P-Frame verwendet werden, sondern theoretisch beliebig viele bereits kodierte Frames gleichzeitig. Dabei spielt es keine Rolle, an welcher Stelle in der Videosequenz die Referenzbilder tatsächlich auftreten. Dadurch können vor allem monoton oder periodisch ablaufende Bewegungen besser kodiert werden.

Verweist ein Block auf mehrere Referenzbilder, ist eine weitere Neuerung in diesem Zusammenhang die Möglichkeit der beliebigen Gewichtung der Referenzblöcke. Dies wird als „Gewichtete Prädiktion“ (*weighted prediction*) bezeichnet und kommt vor allem bei Überblendungen zum Tragen. Desweiteren können B-Frames nun auch selbst als Referenz dienen; die Einschränkung bisheriger Standards, dass P-Frames und B-Frames nur auf I-Frames und P-Frames referenzieren können, wurde bei H.264/AVC abgeschafft.

Auch die Genauigkeit der Bewegungskompensation wurde bei H.264/AVC verbessert: Die Bewegungsvektoren geben die relative Verschiebung eines Makroblocks nun auf $1/4$ Pixel genau an.

Für den Fall, dass sowohl die horizontale als auch die vertikale Komponente des Bewegungsvektors null beträgt, also sich der entsprechende Makroblock zwischen den beiden Frames nicht bewegt hat und sich immer noch an der exakt gleichen Stelle im Bild befindet, wird dieser Makroblock bei H.264/AVC als *skipped macroblock* markiert. Dies hat zur Folge, dass für diesen Makroblock keine Daten kodiert werden, er also bei der Übertragung „übersprungen“ wird, und sich die Datenmenge dadurch weiter reduziert. Der *decoder* verwendet im Fall eines *skipped macroblocks* einfach die Daten aus dem Referenzbild.

Integertransformation und Quantisierung

Sowohl bei der Intrakodierung als auch bei der Interkodierung enthalten die resultierenden Prädiktionsfehler räumliche Redundanzen, welche in der Regel durch eine anschließende Transformation in den Frequenzbereich eliminiert bzw. verringert werden.

Anstatt der sonst üblichen rechenaufwändigen diskreten Kosinustransformation wird bei H.264/AVC eine von der DCT abgeleitete Integertransformation, die Hadamard-

Transformation, verwendet. Der große Vorteil der Integertransformation ist die Beschränkung auf Additionen, Subtraktionen und binäre Verschiebeoperationen, welche mit einer 16 Bit- statt 32 Bit-Arithmetik auskommen, was die benötigte Rechenleistung deutlich verringert und die Hardwareimplementierung vereinfacht.

Bei MPEG-2 wurde die DCT auf Makroblöcke der Größe 8 $\times$ 8 Pixel angewandt, bei H.264/AVC werden 4 $\times$ 4-Blöcke transformiert. Dies reduziert sogenannte Ringing-Artefakte[4] und begünstigt damit die empfundene Videoqualität. Durch die Verwendung der Integertransformation ist außerdem erstmals eine exakte Rücktransformation möglich, was bedeutet, dass jeder standardkonforme *decoder* immer exakt gleiche Bilder rekonstuiert, was ebenfalls ein Novum im Bereich der Videokodierung ist.

Nach der Transformation werden die Koeffizienten verlustbehaftet quantisiert. Die Schrittweite der Quantisierung ist dabei über einen Quantisierungsparameter (QP) flexibel anpassbar. Für den QP ist ein Wert zwischen 0 und 51 erlaubt, wobei höhere Werte eine größere Schrittweite (und damit eine niedrigere Qualität) bedeuten. Die Schrittweite verdoppelt sich, wenn der QP um 6 erhöht wird [12].

Adaptive Entropiekodierung

Statt der üblichen Entropiekodierung (VLC, siehe 2.2.2), welche bei H.264/AVC 'Context Adaptive Variable Length Coding' (CAVLC) genannt wird, ist optional eine leistungsfähigere, jedoch auch rechenintensivere arithmetische Entropiekodierung möglich - das sogenannte 'Context Adaptive Binary Arithmetic Coding' (CABAC).

'Context Adaptive', auf deutsch etwa „kontextabhängige Anpassung", deutet an, dass im Vergleich zu früheren Standards beide Methoden der Entropiekodierung effizienter arbeiten, indem sie sich an die Merkmale und Parameter des zu kodierenden Materials anpassen.

Integrierter Deblocking-Filter

Negative Eigenschaften jedes blockbasierten Videokodierungs-Algorithmus sind Blocking-Artefakte, welche vom Beobachter als besonders störend empfunden werden und daher die subjektive Videoqualität stark beeinträchtigen. Bei allen Vorgängern von H.264/AVC wurde daher oft nach der Dekodierung noch ein externer Deblocking-Filter zur empfundenen Qualitätssteigerung auf das Videomaterial angewandt.

Bei H.264/AVC wurde bereits im Dekodierungsvorgang ein adaptiver Deblocking-Filter integriert. Dieser befindet sich innerhalb der Bewegungskompensations-Schleife [22], was den Vorteil hat, dass die Filterergebnisse direkt für die Bewegungsschätzung verwendet werden können und damit, als positiver Nebeneffekt, die Zuverlässigkeit und Genauigkeit der Interkodierung erhöht wird.

[4] Ringing-Artefakte treten an Kanten mit großen Farb- oder Helligkeitsdifferenzen auf (z. B. bei Text im Bild) und entstehen bei der Transformation in den Frequenzbereich durch ein Über- oder Unterschwingen an der Kante.

Da das Auftreten und die Ausgeprägtheit von Blocking-Artefakten stark mit der Quantisierung korreliert (je feiner die Quantisierung, desto seltener und schwächer treten Blocking-Artefakte auf), werden die Schwellwerte des adaptivenden Deblocking-Filters je nach Quantisierungstiefe angepasst.

Verbesserte Robustheit und Plattformintegrität

Um gegen Datenverluste während der Übertragung gewappnet zu sein, gibt es bei H.264/AVC die Möglichkeit, wichtige Regionen oder auch ganze Slices oder Frames redundant zu kodieren, wobei die redundanten Informationen mit einer niedrigeren Bitrate und damit schlechteren Qualität kodiert werden und nur dann dekodiert werden, wenn die entsprechenden Originalinformationen auf der Dekoderseite aufgrund von Übertragungsfehlern nicht mehr verfügbar sind.

Ein weiteres Merkmal von H.264/AVC, das der Fehler-Robustheit zugute kommt, ist das sogenannte 'Data Partitioning': Darunter versteht man die Unterteilung in bis zu drei Partitionen, die mit unterschiedlicher Priorität übertragen werden. Somit können wichtigere Informationen (z.B. die Bewegungsvektoren) höher priorisiert übertragen werden als unwichtigere Informationen.

Auch können optional sogenannte 'Switching Slices' verwendet werden: Die als SI-Slice ('Switching I-Slice') bzw. SP-Slice ('Switching P-Slice') bezeichneten Slice-Typen wurden eingeführt, um dem *decoder* ein Umschalten zwischen zwei verschiedenen Videoströmen (mit unterschiedlicher Datenrate) zu ermöglichen.

Damit der Codec H.264/AVC für die verschiedensten Anwendungen und Übertragungskanäle (z.B. Blu-Ray, Streaming über Internet, DVB, ...) verwendet werden kann, wurde großer Wert auf die Anpassungsfähigkeit an die jeweilige Plattform gelegt. Dazu wurde der eigentliche Kern des Codecs in eine plattformunabhängige Struktur gepackt, die sogenannte 'Video Coding Layer' (VCL), welche von der umhüllenden 'Network Abstraction Layer' (NAL) an das entsprechende Übertragungsprotokoll angepasst wird. Man kann die NAL daher als Schnittstelle zwischen Codec und der Außenwelt ansehen.

Mehr zu NAL, VCL und der weiteren Struktur von H.264/AVC sowie einen Überblick über die Bitstream-Syntax und -Semantik findet sich im folgenden Kapitel 2.4.

2.4 H.264/AVC Bitstream-Struktur

Dieses Kapitel soll einen groben Überblick über die Struktur des Bitstreams von H.264/AVC kodiertem Video geben; sämtliche Details und Einzelheiten finden sich in der aktuellen ITU-T H.264 Recommendation [8]. Als Referenz für diese Arbeit diente die Version 03/2010.

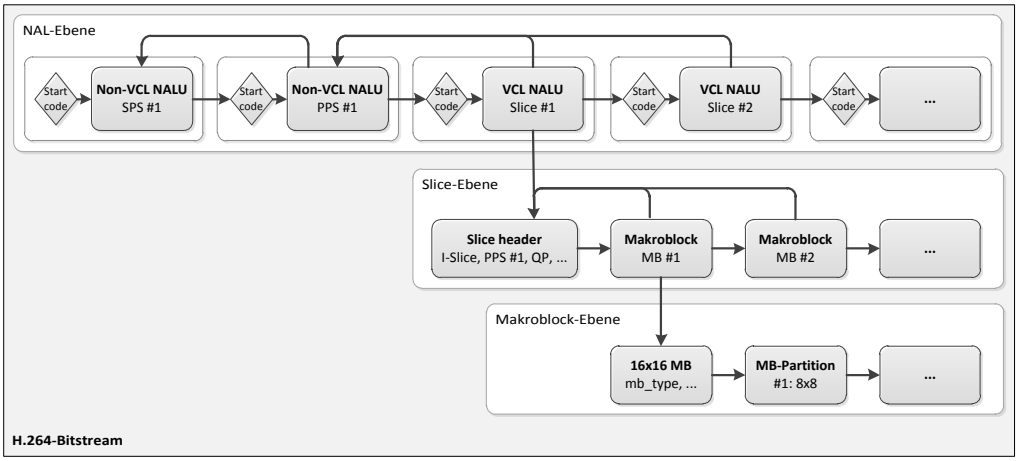


Abbildung 2.7: Struktur des H.264-Bitstreams im *byte stream format* (siehe Anhang B in [15]). Dieser lässt sich in die 3 Schichten 'NAL', 'Slice' und 'Makroblock' unterteilen.

2.4.1 Ebene 1: NAL

Wie im Abschnitt 2.3.2 bereits vorweggenommen, ist die oberste Schicht die sogenannte 'Network Abstraction Layer' (NAL), welche die kodierten Videodaten an das jeweilige Übertragungsprotokoll anpassbar macht. Im Folgenden wird die Einbettung in die verschiedenen Übertragungswege jedoch nicht berücksichtigt, sondern lediglich der „rohe" Bitstream im songenannten *byte stream format* (siehe Anhang B in [15]) betrachtet. Abbildung 2.7 zeigt eine Skizze der Ebenen des H.264-Bitstreams im *byte stream format*.

Der H.264-Bitstream besteht aus Paketen, welche eine ganzzahlige Anzahl an Bytes enthalten und als 'Network Abstraction Layer Units' (NALU) bezeichnet werden. NALUs werden in Non-VCL und VCL unterteilt. Non-VCL NALUs sind normalerweise die ersten Pakete am Anfang des Bitstreams; zu dieser Kategorie gehören die 'Sequence Parameter Sets' (SPS) und 'Picture Parameter Sets' (PPS), welche weiter unten in diesem Kapitel noch näher vorgestellt werden. Weitere Non-VCL NALUs sind unter anderem die für das *decoding* nicht zwingend notwendigen 'Supplemental Enhancement Information' (SEI) und 'Access Unit Delimiter' (AUD), welche aufgrund ihrer Optionalität keine Rolle für diese Arbeit spielen konnten und daher nicht näher erläutert werden. Der Großteil der NALU-Pakete sind VCL NALUs; jede VCL NALU enthält die kodierten Informationen über genau ein Slice.

Oftmals werden mehrere VCL NALUs mit den zugehörigen SEI- und AUD-Paketen zu einer *access unit* (AU) zusammengefasst, wobei eine AU genau einem dekodierten Einzelbild entspricht[5].

[5] Dies macht jedoch nur dann Sinn, wenn jedes Frame in mehrere Slices unterteilt ist und die optionalen SEI- und AUD-Pakete vorhanden sind - ansonsten entspricht eine AU einer VLC NALU.

Im *byte stream format* ist jeder neuen NALU ein 3 Byte langer *start code prefix* vorangestellt, über welchen der *decoder* den Anfang eines neuen Paketes eindeutig identifizieren kann. Die NALU selbst beginnt dann stets mit einem 1 Byte großen *header*, welcher nach einem Prüfbit die Priorität und den Typ der NALU enthält. In Tabelle 2.1 wird eine Übersicht über die wichtigsten NALU Typen gegeben.

Tabelle 2.1: Einige NALU Typen und deren Klassifizierung

nal_unit_type	Inhalt der NALU	Klassifizierung
1-5	Kodiertes Slice	VCL
7	Sequence Parameter Set (SPS)	Non-VCL
8	Picture Parameter Set (PPS)	Non-VCL

Die Parameter Sets SPS und PPS enthalten Informationen und Parameter, die für mehrere Slices konstant bleiben und daher aus Effizienzgründen nur einmal übertragen werden.

Wie aus den Bezeichnungen der Parameter Sets ersichtlich ist, gilt ein SPS für eine sehr große Anzahl an NALUs („Sequenz“) und ein PPS für ein oder mehrere Einzelbilder. Jede VCL NALU referenziert auf genau ein PPS, welches wiederum auf ein SPS verweist.

In den meisten Fällen gibt es pro Videodatei nur ein einziges SPS, das für alle VCL NALUs dieser Sequenz gilt, und oftmals auch nur maximal eine Hand voll PPS. Das SPS enthält daher allgemein gültige Videoinformationen wie z. B. das Encoding-Profil und Level, die Auflösung und die Bittiefe, während in den PPS eher adaptive Parameter für die entsprechenden Slices gesetzt werden können (z. B. die verwendete Methode der Entropiekodierung oder die initialen Quantisierungsparameter). Da ohne die in den Parameter Sets enthaltenen Daten eine Dekodierung des Bitstreams nicht möglich ist, leuchtet es ein, dass sie meist ganz zu Beginn des Datenstroms kodiert sind und oft redundant oder sogar über einen separaten Kanal übertragen werden.

Während Non-VCL NALUs sowohl zahlenmäßig als auch größenmäßig mit dem Umfang einiger Bytes den geringsten Anteil am H.264-Bitstream ausmachen, sind die tatsächlichen Bildinformationen in den VCL NALUs enthalten. Diese lassen sich in zwei weitere Hierarchieebenen gliedern, welche in den folgenden beiden Unterkapiteln 2.4.2 und 2.4.3 vorgestellt werden.

2.4.2 Ebene 2: Slice (VCL)

Die Informationen eines Slices sind wie bereits unter 2.4.1 erwähnt im AVC-Bitstream zu einem VCL NALU-Paket zusammengefasst. Dieses beginnt mit dem *slice header*, welcher wichtige Parameter des jeweiligen Slices enthält. Dies sind z. B. der Slice-Typ (I-Slice, P-Slice, B-Slice), das korrespondierende PPS, das für dieses Slice gilt, sowie der initiale Quantisierungsparameter, der beim Kodieren des Slices verwendet wurde.

Auf den *slice header* folgen dann die eigentlichen Daten, mit denen das entsprechende

Slice bei der Dekodierung rekonstruiert werden kann. Dazu existiert eine Schleife über alle 16 $\times$ 16-Makroblöcke des aktuellen Slices, die abstrakt betrachtet in die dritte Ebene, die sogenannte *macroblock layer*, verzweigt.

2.4.3 Ebene 3: Makroblock

Der erste Parameter, der für jeden 16 $\times$ 16 großen Makroblock im *macroblock layer* kodiert ist, ist dessen Typ (`mb_type`):

Für I-Slices kommen als Typ lediglich die Grundform 16 $\times$ 16, eine Unterteilung in Unterblöcke der Größe 8 $\times$ 8 oder 4 $\times$ 4 sowie die Sonderform I_PCM in Frage. Bei P-Slices und B-Slices enthält der Integerwert `mb_type` neben der Größe des Makroblocks auch die Information, ob er intrakodiert oder interkodiert ist. Für den Fall der Intrakodierung sind die möglichen Blockgrößen identisch zu denen der I-Slices, bei Interkodierung kann jeder Makroblock eine der folgenden Größen annehmen: 16 $\times$ 16, 16 $\times$ 8, 8 $\times$ 16, 8 $\times$ 8, 8 $\times$ 4, 4 $\times$ 8 oder 4 $\times$ 4 (siehe auch Abschnitt 2.3.2). Zudem gibt es den Typ 'Skip' für *skipped macroblocks*.

Weitere für diese Arbeit relevante Informationen, die aus der Makroblock-Ebene gewonnen werden können, sind die Anpassung der Quantisierungsparameter für den jeweiligen Makroblock und im Falle von interkodierten Makroblöcken die Bewegungsvektoren. Die Richtung und Länge der Bewegungsvektoren kann dabei über die im Bitstream kodierten Werte mvd_x und mvd_y berechnet werden, welche die Differenzen zwischen geschätzter und tatsächlicher Länge des Bewegungsvektors in x- und y-Richtung angeben.

Nachdem nun die Grundlagen für das Verständnis der Merkmalsextraktion aus dem H.264-Bitstream geschaffen wurden, kann nun ein Blick auf die im Rahmen dieser Arbeit ausgelesenen Daten und deren Analyse geworfen werden. Zuvor wird jedoch noch ein kurzer Überblick über den Stand der Technik auf diesem Gebiet gegeben.

3 Merkmalsextraktion aus dem H.264-Bitstream

Dieser Abschnitt befasst sich mit der Merkmalsextraktion aus dem Bitstream von H.264-kodiertem Videomaterial. Dazu werden zunächst unter 3.1 verwandte Forschungsarbeiten und Veröffentlichungen aus diesem Bereich vorgestellt und in Kapitel 3.2 die verwendeten Funktionen und Begriffe aus der Statistik sowie deren Berechnung angesprochen, bevor unter 3.3 die im Rahmen dieser Arbeit durchgeführten Software-Implementierungen aufgezeigt und schließlich unter 3.4 alle aus dem H.264-Bitstream extrahierten Parameter aufgeführt und erklärt werden.

3.1 Stand der Technik

Bei der Literaturrecherche im Rahmen dieser Diplomarbeit wurde untersucht, welche veröffentlichten Arbeiten die Thematik der *no-reference*-Qualitätsevaluierung von H.264/AVC-Videosequenzen bereits behandelt haben und inwiefern sie mit dem Ansatz dieser Arbeit vergleichbar sind. Dabei wurde festgestellt, dass der Großteil der Forschungsansätze das bereits dekodierte Videomaterial heranzieht, um die Videoqualität abzuschätzen.

So haben z.B. C. Keimel et al. in [10] einen interessanten Ansatz vorgestellt, welcher charakteristische Merkmale wie z. B. *bluriness*, *blockiness*, *activity* oder *predictability* aus den dekodierten Bildern extrahiert und anhand dieser *features* eine Qualitätsschätzung vornimmt. Einen anderen vielversprechenden Ansatz hatten S. Péchard et al. [13], welcher allerdings leider keine repräsentativen Ergebnisse liefert, da zur Modellkalibrierung und zur Modellvalidierung die gleichen Objekte und Daten verwendet wurden.

O. Sugimoto et al. verfolgten in einer erst kürzlich veröffentlichten Arbeit [19] die Idee einer hybriden Qualitätsmetrik, welche sowohl auf Merkmalen aus dem dekodierten Material als auch auf Bitstream-Features beruht. Hierbei wurden aus dem H.264-Bitstream unter anderem der durchschnittliche Quantisierungsparameter, die Bitrate, die Framerate, die durschnittliche Bewegungsvektorlänge sowie das Verhältnis der verschiedenen Makroblockgrößen ausgelesen und zur Qualitätsbeurteilung verwendet.

Es fanden sich jedoch auch einige Arbeiten und Methoden, die nur anhand von Bitstream-Features eine Qualitätsevaluierung vornahmen. Ein Beispiel dafür war der Artikel *„No-reference estimation of the coding PSNR for H. 264-coded sequences“* von A. Eden [4]. Der Autor stellte darin einen Algorithmus vor, der das PSNR über die

Quantisierungsparameter sowie nicht näher genannte weitere Merkmale aus dem H.264-Bitstream abschätzte.

Der '*feature extractor*' von M. Slanina et al. liest ebenfalls die Quantisierungsparmeter sowie das Verhältnis der verschiedenen Blockgrößen (16×16, 8×8, 4×4) aus und schätzt über diese beiden Bitstream-Features das PSNR ab (Details siehe [17]).

Eine deutlich größere Anzahl an verschiedenen Bitstream-Features extrahierten A. Rossholm et al. Sie bildeten in ihrer Veröffentlichung *„A new video quality predictor based on decoder parameter extraction“* [16] eine Qualitätsmetrik, welche die Bildqualität von QCIF-Videosequenzen (Auflösung: 176×144) über folgende zehn Merkmale aus dem H.264-Bitstream bestimmen sollte:

1. Durchschnittlicher Quantisierungsparameter
2. Bits/Frame
3. Anteil intrakodierter Blöcke
4. Anteil interkodierter Blöcke
5. Anteil übersprungener Blöcke (*skipped blocks*)
6. Framerate
7. Anteil interkodierter 16×16 Blöcke
8. Anteil interkodierter 16×8, 8×16 bzw. 8×8 Blöcke
9. Anteil interkodierter 8×4, 4×8 bzw. 4×4 Blöcke
10. Durchschnittliche Länge der Bewegungsvektoren

Die Arbeit von A. Rossholm et al. wies die größte Ähnlichkeit zum Konzept dieser Arbeit auf und diente damit einerseits als Hauptbezugsquelle und Ideengeber, andererseits als Referenz bei der Auswertung der Ergebnisse, welche später unter Punkt 5 noch vorgestellt werden. Zunächst werden jedoch in Kapitel 3.4 alle im Rahmen dieser Diplomarbeit ausgelesenen Merkmale aufgeführt und die dafür nötigen Implementierungen unter 3.3 gezeigt. Davor werden im folgenden Unterpunkt 3.2 die gebrauchten Begriffe aus der Statistik kurz eingeführt.

3.2 Verwendete Funktionen der Statistik

Da für die Merkmalsextraktion verschiedene Maße der Statistik herangezogen wurden, werden die verwendeten Funktionen und deren Berechnungsgrundlage in diesem Kapitel kurz erläutert.

Mittelwert

Der Mittelwert ist der Durchschnitt von n Werten $x_1, x_2, \ldots, x_n$ und kann folgendermaßen berechnet werden:

$$\text{Mittelwert } x_{avg} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n} \tag{3.1}$$

Die korrekte mathematische Bezeichnung des Mittelwertes ist 'arithmetisches Mittel'.

Median

Hat man eine sortierte Liste $x_1 < x_2 < \cdots < x_n$ und ist n ungerade, so ist der Median dieser Liste anschaulich gesehen der Wert in der Mitte der Liste (Zentralwert). Ist n gerade, so berechnet sich der Median über den Mittelwert der beiden mittigen Listeneinträgen:

$$\text{Median } x_{med} = \begin{cases} \frac{1}{2}(x_{\frac{n}{2}} + x_{\frac{n}{2}+1}) & \text{falls } n \text{ gerade,} \\ x_{\frac{n+1}{2}} & \text{falls } n \text{ ungerade.} \end{cases} \tag{3.2}$$

Der Median hat im Vergleich zum Mittelwert den Vorteil, robuster gegenüber Ausreißern zu sein. So ist bespielsweise für die Liste $[1, 3, 4, 6, 21]$ der Mittelwert $x_{avg} = 7$ und der Median $x_{med} = 4$.

Standardabweichung

Die Standardabweichung, oft auch als Varianz bezeichnet, ist das Maß für die Streuung der Werte einer Liste $x_1, x_2, \ldots, x_n$ um ihren Mittelwert und wird nach folgender Gleichung berechnet:

$$\text{Standardabweichung } x_{sd} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - x_{avg})^2}{n-1}} \tag{3.3}$$

Dabei ist x_{avg} der Mittelwert der Liste $x_1, x_2, \ldots, x_n$. Allgemein gesagt gilt: Je geringer die Standardabweichung, desto ähnlicher sind sich die Werte der untersuchten Liste.

p-Quantil

Das p-Quantil ist ein Lagemaß der Statistik, welches die sortierte Liste $x_1 < x_2 < \cdots < x_n$ in zwei Bereiche teilt: Links des p-Quantils liegen p Prozent der Listeneinträge, deren Werte allesamt kleiner oder gleich des p-Quantils sind; rechts davon liegen die restlichen (größeren) Listeneinträge. Das 10-Quantil der Liste $x_1 < x_2 < \cdots < x_n$ ist also beispielsweise der Eintrag x_i, der größer (oder gleich) ist als 10 Prozent der Liste.

Eine Sonderform des *p*-Quantils ist der Median, welcher auch als 50-Quantil bezeichnet werden könnte.

Minimum/Maximum

Diese beiden selbsterklärenden Maße der Statistik erfassen den Minimalwert bzw. Maximalwert einer Liste. Für das Minimum x_{min} bzw. Maximum x_{max} der Liste $x_1, x_2, \ldots, x_n$ gilt:

$$\text{Minimum } x_{min} \leq x_i \quad \forall i \in \{1 \ldots n\} \tag{3.4}$$

$$\text{Maximum } x_{max} \geq x_i \quad \forall i \in \{1 \ldots n\} \tag{3.5}$$

3.3 Implementierung

Wie unter 4.1 noch ausführlich verdeutlicht wird, war das primäre Ziel dieser Arbeit zunächst, möglichst viele charakteristische Merkmale aus dem H.264-Bitstream auszulesen. Dabei spielte es vorerst keine Rolle, ob jedes einzelne Merkmal überhaupt einen (und falls ja, welchen) Einfluss auf die Videoqualität hat. Eine lineare Abhängigkeit der einzelnen Bitstream-Features war hierbei auch nicht von Nachteil, sondern im Gegenteil sogar erwünscht.

Nachdem sich herausstellte, dass eine völlig eigenständige Software-Implementierung eines '*feature extractors*' den zeitlichen Umfang dieser Diplomarbeit deutlich sprengen würde, wurde auf die H.264/AVC Referenzdecoder-Implementierung zurückgegriffen und diese soweit modifiziert, dass der eigentliche Dekodierungsprozess nicht mehr stattfand, sondern lediglich die benötigten Bitstream-Features ausgelesen wurden. Die in der Programmiersprache C geschriebene H.264/AVC Referenzsoftware ist unter [18] frei verfügbar; verwendet wurde die zum Zeitpunkt der Implementierung aktuelle Version JM 17.2.

Im folgenden Kapitel 3.3.1 wird ein Überblick über die Struktur des Referenzdecoders in Pseudocode gegeben und anschließend werden unter 3.3.2 die wesentlichen Anpassungen und eigenen Implementierungen zusammengefasst.

3.3.1 Struktur der H.264/AVC Referenzsoftware

Lädt man das H.264/AVC Referenzsoftware-Projekt, finden sich in der Projektmappe die vier Teilprojekte 'ldecod', 'lencod', 'rtp_loss' sowie 'rtpdump'. Relevant für diese Arbeit war ausschließlich das Projekt 'ldecod', welches die Referenzdecoder-Implementierung darstellt. In diesem Teilprojekt ist neben zahlreichen Header-Dateien (.h) und Quellcode-Dateien (.c) in C-Code auch eine Konfigurationsdatei `decoder.cfg` vorhanden, in der wichtige Parameter für die Ausführung des kompilierten Programms (wie z. B. Name und Pfad der Input-Datei) gesetzt werden.

Nachfolgend wird nun ausgehend vom Haupteinstiegspunkt des Referenzdecoders, welcher sich in der Quellcode-Datei `decoder_test.c` befindet, ein Pseudocode-Überblick über die Verzweigung der für diese Arbeit wichtigen Funktionen gegeben. Hinter jeder Funktion wird außerdem die Quellcode-Datei angegeben, in der sie zu finden ist (im Stil `Funktion() @ Quellcode-Datei`).

Struktur des H.264/AVC Referenzdecoders 'ldecod' (JM 17.2)

```
main() @ decoder_test.c
{
  while more_data_in_Bitstream do      //MAIN LOOP
    DecodeOneFrame() @ ldecod.c
    {
      decode_one_frame() @ image.c
      {
        while current_slice_is_part_of_current_frame do
          read_new_slice() @ image.c
          {
            repeat
              read_next_nalu @ nalu.c
              if nalu_type == SPS then
                ProcessSPS() @ parset.c
                → InterpretSPS @ parset.c
              end if
              if nalu_type == PPS then
                ProcessPPS() @ parset.c
                → InterpretPPS @ parset.c
              end if
              if nalu_type == SLICE then
                FirstPartOfSliceHeader() @ header.c
                UseParameterSet() @ parset.c
                RestOfSliceHeader() @ header.c
              end if
            until end_of_stream
          }
        end while
        for all_slices_of_current_frame do
          decode_slice() @ image.c
          {
            decode_one_slice() @ image.c
            {
              while not_end_of_slice do      //MB LOOP
```

```
                read_one_macroblock_X_slice() @ macroblock.c
                //X ∈ {I, P, B}
                {
                  update_qp() @ macroblock.c
                  interpret_mb_mode_X() @ macroblock.c
                  //X ∈ {I, P, B}
                  if slice_type != I then
                    read_motion_info_from_NAL_p/b_slice()
                    @ macroblock.c
                    {
                      readMBMotionVectors @ macroblock.c
                    }
                  end if
                }
              end while
            }
          }
        end for
      }
    }
  end while
}
```

3.3.2 Wesentliche Modifizierungen des Referenzdecoders

Damit die gewünschten Parameter und Merkmale (Bitstream-Features) über die H.264/AVC Referenzdecoder-Implementierung ausgelesen und berechnet werden konnten, wurde zunächst in der Header-Datei `global.h` eine Struktur `features` angelegt. Diese Struktur enthält alle für die eigenen Implementierungen benötigten Variablen und Listen (*arrays*) und wurde anschließend global unter dem Namen `feat` deklariert.

Zur Kalkulation des Mittelwertes, des Medians, der Standardabweichung, des *p*-Quantils sowie des Minimums und Maximums der *feature-arrays* wurden die sechs Funktionen `average()`, `median()`, `sdeviation()`, `quantil()`, `findmin()` und `findmax()` in der Quellcode-Datei `decoder_test.c` implementiert. Bis auf die Funktion `median()`, welche eine Abwandlung der 'Quick select'-Methode aus [5] darstellt, handelt es sich dabei um selbst geschriebenen Code nach den Berechnungsgrundlagen aus 3.2.

Alle finalen Berechnungen sowie die Konsolen-Ausgabe der Bitstream-Features und des geschätzten Qualitätswertes q_x (siehe Kapitel 4.1) erfolgten in der `main()`-Funktion (Datei `decoder_test.c`) nach Analyse des gesamten Bitstreams, also unmittelbar vor Beendigung des Programms. In der gleichen Funktion wurde nach der Durchführung der

multivariaten Datenanalyse (Abschnitt 4) der zur Qualitätsprognose benötigte Gewichtungsvektor $\vec{b}$ (siehe 4.1) hinterlegt.

In der Code-Datei `image.c` wurden am Ende der Funktion `decode_one_slice()` nach der Schleife über alle Makroblöcke des Slices die Berechnungen der Variablen für das jeweilige Slice durchgeführt und einige Merkmale und Parameter des Slices über die Konsole ausgegeben.

Alle weiteren Codeanpassungen, die zur Extraktion der Bitstream-Features in der H.264/AVC Referenzdecoder-Software vorgenommen wurden, werden im nächsten Abschnitt 3.4 unter den jeweiligen Merkmalen erwähnt.

3.4 Übersicht der ausgelesenen Bitstream-Features

Insgesamt wurden im Rahmen dieser Arbeit über die im vorangegangenen Abschnitt 3.3 vorgestellte modifizierte H.264/AVC Referenzsoftware folgende 64 Parameter aus dem Bitstream der H.264-kodierten Videosequenzen ausgelesen:

- **Profile**

  Das beim *encoding* verwendete H.264-Profil wird im Sequence Parameter Set (SPS) als Integerzahl kodiert an den *decoder* übermittelt. Je höher der Wert, desto „hochwertiger“ ist auch das Profil (77 entspricht beispielsweise dem Main-Profil, 100 steht für das High-Profil). Diese Ziffer wird als erstes Merkmal aus der Funktion `InterpretSPS()` in der Codedatei `parset.c` ausgelesen.

- **Level**

  Analog zum Profil findet sich auch das Level im SPS als Integerwert wieder (40 = Level 4.0, 51 = Level 5.1, etc.). Dieser wurde ebenfalls aus der Funktion `InterpretSPS()` in der Quellcode-Datei `parset.c` ausgelesen.

- **Entropy**

  Die verwendete Methode der Entropiekodierung (CAVLC oder CABAC) ist in den Picture Parameter Sets (PPS) als `entropy_coding_mode_flag` kodiert. Dieser *flag* hat für CAVLC den Wert 0 und für CABAC den Wert 1. Sind für eine Videosequenz mehrere PPS mit unterschiedlichen Entropiekodierungs-Methoden vorhanden [1], so wird der Mittelwert gebildet. Die Implementierung im Referenzdecoder erfolgte in der Funktion `InterpretPPS()` (`parset.c`).

- **$\mathbf{kbit_{avg}}$**

  Mittelwert der Kilobits, die für die Kodierung eines Slice in Anspruch genommen wurden. Hierfür wurden die Bytes gezählt, die eine NALU umfasst, und im Falle

[1] Bei den vorhandenen Testsequenzen war jedoch ausschließlich jeweils ein PPS pro Sequenz kodiert.

einer VCL-NALU in Kilobit umgerechnet und in das entsprechende *bitrate-array* geschrieben, wovon dann der Mittelwert gebildet werden konnte. Für die Berechnung wurden wie gesagt nur VCL-NALUs berücksichtigt. Die Anzahl der Bytes fand sich in der Funktion `read_next_nalu` (`nalu.c`).

- **kbit_{med}**

  Median der Kilobit pro Slice (Zentralwert des *bitrate-arrays*) (siehe oben).

- **kbit_{sd}**

  Standardabweichung der Kilobit pro Slice (siehe oben).

- **$\text{kbit}_{\text{10-Q}}$**

  10-Quantil der Kilobit pro Slice (siehe oben).

- **$\text{kbit}_{\text{90-Q}}$**

  90-Quantil der Kilobit pro Slice (siehe oben).

- **kbit_{min}**

  Minimalwert der Kilobit pro Slice (siehe oben).

- **kbit_{max}**

  Maximalwert der Kilobit pro Slice (siehe oben).

- **QP_{avg}**

  Mittelwert der Quantisierungsparameter. Dazu wurde zunächst für jedes Slice der durchschnittliche Quantisierungsparameter bestimmt, indem der initiale Quantisierungsparameter aus dem *slice-header* (Funktion `RestOfSliceHeader()`, Datei `header.c`) ausgelesen und eine eventuelle Anpassung des Quantisierungsparameters auf Makroblock-Ebene (Funktion `update_qp()`, Datei `macroblock.c`) eingerechnet wurde. Falls es keine Veränderung auf Makroblock-Ebene gab, entsprach der durchschnittliche Quantisierungsparameter des Slices dem initialen Quantisierungsparameter im *slice-header*. Die durchschnittlichen Quantisierungsparameter jedes Slices wurden in ein davor vorgesehenes *qp-array* geschrieben, um analog zur Bitrate auch die anderen statistischen Funktionen anwenden zu können.

- **QP_{med}**

  Median des *qp-arrays* (siehe oben).

- **QP_{sd}**

  Standardabweichung der durchschnittlichen Quantisierungsparameter pro Slice (siehe oben).

- **$QP_{10\text{-}Q}$**

  10-Quantil der durchschnittlichen Quantisierungsparameter pro Slice (siehe oben).

- **$QP_{90\text{-}Q}$**

  90-Quantil der durchschnittlichen Quantisierungsparameter pro Slice (siehe oben).

- **QP_{min}**

  Minimalwert des *qp-arrays* (siehe oben).

- **QP_{max}**

  Maximalwert des *qp-arrays* (siehe oben).

- **MVl_{avg}**

  Mittelwert der durchschnittlichen Bewegungsvektorlänge pro interkodiertem Slice. Die Berechnung der Länge eines Bewegungsvektors erfolgte über die im *macroblock-layer* (Funktion `readMBMotionVectors()`, Datei `macroblock.c`) ausgelesenen Werte mvd_x und mvd_y, welche die Abweichungen der geschätzten Länge von der tatsächlichen Länge des Bewegungsvektors in x- und y-Richtung angeben (siehe auch Abschnitt 2.4.3). Nach Addition der geschätzten Werte mv_pred_x und mv_pred_y erhielt man den tatsächlichen Betrag des Bewegungsvektors in x- und y-Richtung:

  $$MVl_i = mv\_pred_i + mvd_i \quad mit\ i \in \{x, y\} \tag{3.6}$$

  Anschließend konnte über Gleichung 3.7 die Bewegungsvektorlänge *MVl* berechnet werden.

  $$MVl = \sqrt{MVl_x^2 + MVl_y^2} \tag{3.7}$$

  Für jedes interkodierte Slice wurde von allen berechneten Bewegungsvektoren *MVl* der Mittelwert gebildet und dieser in das *mv-array* geschrieben. Die Länge dieses *arrays* entsprach somit der Anzahl an interkodierten Slices .

- **MVl_{med}**

  Median des *mv-arrays* (siehe oben).

- **MVl_{sd}**

  Standardabweichung der durchschnittlichen Bewegungsvektorlänge pro interkodiertem Slice (siehe oben).

- **$MVl_{10\text{-}Q}$**

  10-Quantil des *mv-arrays* (siehe oben).

- **$MVl_{90\text{-}Q}$**

  90-Quantil des *mv-arrays* (siehe oben).

- **MVl_{min}**

  Minimalwert des *mv-arrays* (siehe oben).

- **MVl_{max}**

  Maximalwert des *mv-arrays* (siehe oben).

- **$MVMn_{avg}$**

  Für jedes interkodierte Slice wurde nicht nur der Mittelwert aller Bewegungsvektoren berechnet, sondern auch das Minimum und Maximum pro Slice in den Listen *mv_min-array* und *mv_max-array* gespeichert. Der Mittelwert des *mv_min-arrays* wurde als Parameter $MVMn_{avg}$ extrahiert.

- **$MVMn_{med}$**

  Median des *mv_min-arrays* (siehe oben).

- **$MVMn_{sd}$**

  Standardabweichung des *mv_min-arrays* (siehe oben).

- **$MVMn_{10\text{-}Q}$**

  10-Quantil des *mv_min-arrays* (siehe oben).

- **$MVMn_{90\text{-}Q}$**

  90-Quantil des *mv_min-arrays* (siehe oben).

- **$MVMn_{max}$**

  Maximalwert des *mv_min-arrays* (siehe oben).

- **$MVMx_{avg}$**

  Mittelwert des *mv_max-arrays* (siehe oben).

- **$MVMx_{med}$**

  Median des *mv_max-arrays* (siehe oben).

- **$MVMx_{sd}$**

  Standardabweichung des *mv_max-arrays* (siehe oben).

- **$MVMx_{10\text{-}Q}$**

  10-Quantil des *mv_max-arrays* (siehe oben).

- **$MVMx_{90\text{-}Q}$**

  90-Quantil des *mv_max-arrays* (siehe oben).

- **$MVMx_{min}$**

  Minimalwert des *mv_max-arrays* (siehe oben).

- **$MVMx_{max}$**

  Maximalwert des *mv_max-arrays* (siehe oben). Der Wert entspricht dem globalen Maximum, also dem längsten Bewegungsvektor in der gesamten Videosequenz.

- **MVd_{avg}**

  Mittelwert der durchschnittlich im *macroblock-layer* kodierten mvd_x- und mvd_y-Werte (siehe oben oder unter 2.4.3). Dazu wurden die beiden Raumrichtungen *x* und *y* der Einfachheit halber nicht getrennt beobachtet. Die im *mvd-array* abgespeicherten Werte spiegeln somit die Genauigkeit der Bewegungsschätzung wider (je niedriger die Werte, desto genauer war die Prädiktion).

- **MVd_{med}**

  Median des *mvd-arrays* (siehe oben).

- **MVd_{sd}**

  Standardabweichung des *mvd-arrays* (siehe oben).

- **$MVd_{10\text{-}Q}$**

  10-Quantil des *mvd-arrays* (siehe oben).

- **$MVd_{90\text{-}Q}$**

  90-Quantil des *mvd-arrays* (siehe oben).

- **MVd_{min}**

  Minimalwert des *mvd-arrays* (siehe oben).

- **MVd_{max}**

  Maximalwert des *mvd-arrays* (siehe oben).

- **$MVdMx_{avg}$**

  Neben dem Mittelwert der Parameter mvd_x und mvd_y wurde auch deren Maximum pro Slice ausgewertet. Dabei wurde ebenfalls keine Unterscheidung zwischen *x* und *y* gemacht. Die Maximalwerte wurden in der Liste *mvd_max-array* gespeichert.

- **$MVdMx_{med}$**

  Median des *mvd_max-arrays* (siehe oben).

- **$\mathbf{MVdMx_{sd}}$**

  Standardabweichung des *mvd_max-arrays* (siehe oben).

- **$\mathbf{MVdMx_{10\text{-}Q}}$**

  10-Quantil des *mvd_max-arrays* (siehe oben).

- **$\mathbf{MVdMx_{90\text{-}Q}}$**

  90-Quantil des *mvd_max-arrays* (siehe oben).

- **$\mathbf{MVdMx_{min}}$**

  Minimalwert des *mvd_max-arrays* (siehe oben).

- **$\mathbf{MVdMx_{max}}$**

  Maximalwert des *mvd_max-arrays* (siehe oben), also der Wert der größten Abweichung für die gesamte Videosequenz.

- **$\mathbf{qpd_{avg}}$**

  Wie weiter oben in diesem Kapitel und unter 2.4.3 bereits erwähnt, konnte der Quantisierungsparameter, der im *slice-header* für das aktuelle Slice initialisiert wurde, auf Makroblock-Ebene optional für jeden Makroblock angepasst werden, um z. B. Bildbereiche mit besonders vielen Details in einem ansonsten detailarmen Einzelbild genauer kodieren zu können.

  Die Differenz zwischen durchschnittlichem Quantisierungsparameter (gespeichert im *qp-array*) und initialem Quantisierungsparameter wurde für jedes Slice berechnet und der Mittelwert dieser Differenzen im Parameter qpd_{avg} ausgelesen.

- **% qpd**

  Bei vielen Slices ist der initiale Quantisierungsparameter allerdings für das gesamte Slice gültig und wird im *macroblock-layer* nicht mehr verändert; der Prozentsatz der Slices, bei denen der Quantisierungsparameter konstant bleibt, wurde in diesem Parameter festgehalten.

- **% I-Slices**

  Prozentualer Anteil der I-Slices (und SI-Slices, falls vorhanden) an der Gesamtzahl an Slices. Um den Typ des jeweiligen Slices zu erhalten, wurde die Variable `slice_type` im *slice-header* ausgewertet. Die Implementierung in der modifizierten Referenzdecoder-Software erfolgte in der Funktion `FirstPartOfSliceHeader()` (Quellcode-Datei `header.c`).

- **% P-Slices**

  Prozentualer Anteil der P-Slices (und SP-Slices, falls vorhanden).

- **% B-Slices**

  Prozentualer Anteil der B-Slices.

- **% Intra**

  Prozentualer Anteil intrakodierter Makroblöcke an der Gesamtzahl der Makroblöcke. Bestimmt wurde dieser Parameter wie auch alle nachfolgenden Bitstream-Features über den Wert von `mb_type` im *macroblock-layer*; die Implementierung erfolgte in den Funktionen `interpret_mb_mode_X()` (mit X $\in \{I, P, B\}$), welche allesamt in der Quellcode-Datei `macroblock.c` zu finden sind.

- **% Inter**

  Prozentualer Anteil interkodierter Makroblöcke an der Gesamtzahl der Makroblöcke.

- **% Skip**

  Prozentualer Anteil „leerer“ (0 Bit) Makroblöcke (*'skipped macroblocks'*, siehe Kapitel 2.3.2) an der Gesamtzahl der Makroblöcke.

- **% I_**16×16

  Prozentualer Anteil intrakodierter 16×16-Makroblöcke an der Gesamtzahl der Makroblöcke.

- **% I_**8×8

  Prozentualer Anteil intrakodierter 8×8-Makroblöcke an der Gesamtzahl der Makroblöcke.

- **% I_**4×4

  Prozentualer Anteil intrakodierter 4×4-Makroblöcke an der Gesamtzahl der Makroblöcke.

- **% P_**8×8

  Prozentualer Anteil der interkodierten Makroblöcke, die in mehrere Unterpartitionen der Größe 16×8, 8×16, 8×8 oder kleiner aufgespalten wurden. Entspricht der Differenz von allen Makroblöcken minus der Anzahl an „reinen“ 16×16-Makroblöcken.

  Anmerkung zur Nomenklatur: Das Prefix 'P_' steht nicht für P-Slice, sondern allgemein für Interkodierung, umfasst also sowohl die Makroblöcke von P-Slices als auch von B-Slices. Analog dazu repräsentiert das Prefix 'I_' einen intrakodierten Makroblock - unabhängig vom Slice-Typ.

- **% P_**4×4

  Prozentualer Anteil der interkodierten 8×8-Blöcke, die noch weiter unterteilt wurden (in Unterblöcke der Größe 8×4, 4×8 oder 4×4). Entspricht der Differenz von allen 8×8-Blöcken minus der Anzahl an „reinen“ 8×8-Blöcken.

Auch wenn natürlich erwähnt werden muss, dass die im Kapitel 3.1 vorgestellten Arbeiten, insbesondere [16], sehr hilfreiche Denkanreize und Ideen zur Auswahl charakteristischer Merkmale aus dem H.264-Bitstream gaben, wurden einige dieser Bitstream-Features in dieser Art bisher in der Literatur noch nicht aufgeführt. So wurden beispielsweise die Parameter 'qpd_{avg}', '% qpd' oder '$MVdMx_{avg}$' in keinem der bei der Literaturrecherche gefundenen vergleichbaren Forschungsprojekte berücksichtigt.

Der nächste Abschnitt 4 befasst sich nun damit, wie aus dieser Vielzahl an Parametern mithilfe der multivariaten Datenanalyse die geeigneten Bitstream-Features selektiert wurden und das mathematische Modell zur Qualitätsevaluierung gebildet wurde.

4 Multivariate Datenanalyse der Bitstream-Features

Dieses Kapitel beschreibt den zweiten Meilenstein dieser Diplomarbeit, die multivariate Datenanalyse. Zunächst wird unter 4.1 in die Grundlagen und Theorie der multivariaten Datenanalyse eingeführt, bevor die Modellierungsphase in Kapitel 4.2 vorgestellt wird.

4.1 Theorie der multivariaten Datenanalyse

„Beware of wishful thinking!“

Dieses Zitat aus dem Buch *„Multivariate Analysis of Quality: An Introduction“* von H. und M. Martens [11] beschreibt treffend den Grundgedanken, der hinter der multivariaten Datenanalyse steckt. Anstatt sich bei der Frage, welche Parameter und Variablen eines Objektes einen Einfluss auf die Qualität haben, von seinen Gefühlen und Vorurteilen leiten zu lassen und dadurch die vorhandenen Daten einzuschränken und eventuell aussagekräftige Parameter vorab auszuschließen, wird diese Frage bei der multivariaten Datenanalyse von ihr selbst beantwortet.

Um eine Qualitätsevaluierung mit der multivariaten Datenanalyse durchzuführen, sollte man daher zunächst möglichst viele Merkmale (= Parameter bzw. *features*) finden, die die zu untersuchenden Objekte charakterisieren. Abhängige und redundante Parameter sind dabei sogar vorteilhaft und stabilisieren das berechnete Modell (siehe weiter unten). Auch wenn man sich wie gesagt keine genauen Gedanken darüber machen sollte, ob und in wieweit die Merkmale überhaupt mit der Qualität korrelieren, sollten diese natürlich dennoch „sinnvoll sein“ und ein Zusammenhang zur Qualität der Objekte prinzipiell denkbar sein. Zudem sollten die Testobjekte der zu untersuchenden Materie (in unserem Fall waren dies 32 unterschiedlich kodierte H.264-Videosequenzen) im Idealfall die gesamte Bandbreite an möglichen Variationen abdecken.

Hat man eine Vielzahl charakterisierender Merkmale der Testobjekte gesammelt und die Qualität der Testobjekte bereits subjektiv beurteilt, läuft die Qualitätsevaluierung mit der multivariaten Datenanalyse nach folgendem Schema ab:

Die Qualitätswerte $q_1, q_2, \ldots, q_m$ der m Testobjekte werden durch den Qualitätsvektor $\mathbf{y}$ repräsentiert, die n verschiedenen Merkmale $f_1, f_2, \ldots, f_n$ bilden die Spalten der $m \times n$-Matrix $\mathbf{X}$, in dessen m Zeilen die Testobjekte 1 ... m eingetragen sind.

Gesucht ist nun der Gewichtungsvektor $\mathbf{b} = [b_0, b_1, ..., b_n]^T$, der nach Gleichung 4.1 mit der Feature-Matrix **X** multipliziert werden muss, um den Qualitätsvektor **y** in bestmöglicher Näherung zu erhalten:

$$\underbrace{\begin{bmatrix} q_1 \\ \vdots \\ q_m \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} 1 & f_{1,1} & \cdots & f_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & f_{m,1} & \cdots & f_{m,n} \end{bmatrix}}_{X} \cdot \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}}_{b} \tag{4.1}$$

Den Qualitätsvektor **y** und die Feature-Matrix **X** kann man somit als Input-Variablen der multivariaten Datenanalyse betrachten, während der Gewichtungsvektor **b** den Output darstellt. Der Gewichtungsvektor **b**, dessen Elemente $b_0 ... b_n$ auch als Regressionskoeffizienten bezeichnet werden, stellt die Beziehung zwischen den Merkmalen und der Qualität dar und wird in einem komplexen Modellierungsprozess kalkuliert. Dieser Prozess wird als 'Bi-Linear Modelling' (BLM) bezeichnet und besteht im Wesentlichen aus einer 'Partial Least Squares Regression' (PLSR), welche wiederum auf einer 'Hauptkomponentenanalyse' (PCA) basiert. Die für diese Arbeit wichtigen Grundlagen der beiden Werkzeuge werden in den nachfolgenden beiden Unterkapiteln 4.1.1 und 4.1.2 kurz eingeführt, ohne dabei ins mathematische Detail zu gehen; alle Einzelheiten des kompletten mathematischen Modellierungsprozesses der multivariaten Datenanalyse finden sich in [11].

Ausreißer und unbrauchbare Merkmale werden beim Modellierungsprozess der multivariaten Datenanalyse erkannt und können für das endgültige Modell nicht berücksichtigt werden.

Wurde schließlich der passende Gewichtungsvektor **b** gefunden, der die Beziehung zwischen **X** und **y** bestmöglich rekonstruiert, kann in Zukunft mit dessen Hilfe über die Gleichung 4.2 die Qualität q_x eines unbewerteten Objekts **X** (in unserem Fall einer bisher unbewerteten Videosequenz) prognostiziert werden.

$$q_x = b_0 + b_1 \cdot f_{x,1} + b_2 \cdot f_{x,2} + \cdots + b_n \cdot f_{x,n} \tag{4.2}$$

$f_{x,1}, f_{x,2}, ..., f_{x,n}$ sind dabei die n extrahierten Merkmale des unbewerteten Objekts.

4.1.1 Hauptkomponentenanalyse (PCA)

Die Hauptkomponentenanalyse bzw. *Principal Component Analysis* (PCA) führt eine Dimensionsreduktion der Input-Matrix **X** durch. Dazu werden die in der Regel zahlreichen Eingangsvariablen $f_1, f_2, ..., f_n$ zu einigen wenigen Hauptkomponenten, den k *principal components* (PCs), zusammengefasst ($n > k$). Diese PCs sind sogenannte latente Variablen, da sie nicht direkt aus dem Objekt gewonnen werden können, sondern über eine Linearkombination der Merkmale $f_1, f_2, ..., f_n$ des Objekts berechnet werden. Man kann die Hauptkomponenten somit bildlich auch als die Aufsummierung der redundanten bzw. linear abhängigen Informationen in den Eingangsvariablen betrachten.

Die Beziehung zwischen einer Hauptkomponente *a* und den *m* Objekten wird durch den Spaltenvektor $\mathbf{t_a} = \left[t_{1,a}, \cdots, t_{m,a}\right]^T$ ausgedrückt, dessen Elemente als *scores* bezeichnet werden. Der Zeilenvektor $\mathbf{p_a} = \left[p_{a,1}, ..., p_{a,n}\right]$ enthält die *loadings*, welche die Gewichtungen der *n* Eingangsvariablen in der Hauptkomponente *a* darstellen.

Fasst man für alle *k* PCs deren Score-Vektoren zur Matrix $\mathbf{T_k} = [t_1, ..., t_k]$ und deren Loading-Vektoren zur Matrix $\mathbf{P_k} = \left[p_1^T, ..., p_k^T\right]$ zusammen, lässt sich das Modell der Input-Matrix **X** bei der Hauptkomponentenanalyse mit *k* PCs durch Gleichung 4.3 formulieren:

$$X = T_k \cdot P_k^T + E_k \tag{4.3}$$

$\mathbf{E_k}$ ist dabei die Residuenmatrix, die anschaulich betrachtet den Modellfehler, also die Abweichung der modellierten Daten von den tatsächlichen Daten beschreibt. Je besser die Eingangsdaten der Matrix **X** durch die *scores* und *loadings* ausgedrückt werden können (mit anderen Worten: Je passender das Modell der Hauptkomponentenanalyse ist), desto kleiner sind die Werte der Residuenmatrix $\mathbf{E_k}$.

Die Hauptkomponentenanalyse führt zu einer kompakteren, stabileren und einfacheren Modellierung der Daten als die ursprüngliche, *n*-dimensionale Datenmenge der Input-Matrix **X**. Zudem werden die Eingangsvariablen klassifiziert und Ausreißer erkannt, wodurch unbrauchbare Variablen ersichtlich werden und für zukünftige Modelle ausgeschlossen werden können. Je nach Variation in den Merkmalsdaten sind meist zwischen 2 und 5 PCs ausreichend, um alle Eingangsvariablen $f_1, f_2, ..., f_n$ hinreichend zu beschreiben. Werden zu wenige Hauptkomponenten verwendet, ist deren Beschreibung von **X** unzureichend - wählt man hingegen zu viele PCs, so ist die Abhängigkeit des Modells von den Kalibrierungsdaten zu hoch, d.h. das Modell ist zu speziell auf die entsprechenden Testobjekte angepasst und nur bedingt für andere Objekte gültig.

4.1.2 Partial Least Squares Regression (PLSR)

Das allgemeine Ziel einer Regression ist es, für zwei Datenmatrizen **X** und **Y** das Modell $Y = f(X)$ zu erstellen.

Mit Hilfe der 'Partial Least Squares Regression' wird bei der multivariaten Datenanalyse die Beziehung zwischen der Feature-Matrix **X** und dem Qualitätsvektor **y** gesucht, um später über diese Beziehung (welche durch den Gewichtungsvektor **b** beschrieben wird) die Qualität aus den Merkmalen vorhersagen zu können.

Vereinfacht gesagt wird bei der PLSR eine PCA der Feature-Matrix **X** durchgeführt, allerdings mit dem wesentlichen Unterschied, dass dabei als zweite Input-Variable neben der Matrix **X** auch der Qualitätsvektor **y** in die Bestimmung der Hauptkomponenten einbezogen wird, um zu gewährleisten, dass die PCs auch relevant für die Qualitätsbeurteilung sind. Dieser Punkt macht das Regressionsmodell der PLSR im Vergleich zum Modell der PCA einfacher zu interpretieren und statistisch betrachtet verlässlicher [11].

Analog zur PCA beschreibt bei der PLSR die Score-Matrix $\mathbf{T_k}$ die Beziehung zwischen den *k* Hauptkomponenten und den Objekten und die Loading-Matrix $\mathbf{P_k}$ die Gewichtungen

der Merkmale von **X** in den *k* Hauptkomponenten. Ergänzend durch die Residuenmatrix $\mathbf{E_k}$ gilt demnach weiterhin das Modell aus Gleichung 4.3, um die Feature-Matrix **X** durch die Hauptkomponentendarstellung auszudrücken.

Bei der PLSR kommt nun jedoch noch ein Loading-Vektor $\mathbf{q_k}$ ins Spiel, welcher die Beziehung zwischen dem Qualitätsvektor **y** und den *k* Hauptkomponenten beschreibt:

$$y = T_k \cdot q_k^T + f_k \tag{4.4}$$

Der Residuenvektor $\mathbf{f_k}$ enthält dabei analog zu $\mathbf{E_k}$ den Restfehler des Modells und ist für eine aussagekräftige Beziehung zwischen **X** und **y** vernachlässigbar gering. Die Gleichungen 4.3 und 4.4 kann man nun über einige Matrizenmultiplikationen in die gewünschte Form $y = X \cdot b + f_k$ bringen (siehe Gleichung 4.1), wobei die Regressionskoeffizienten von **b** aus $\mathbf{P_k}$ und $\mathbf{q_k}$ berechnet werden können.

Zusammenfassend geben die Gleichungen 4.5 nochmal einen Überblick über das Kalibrierungsmodell der PLSR.

$$\begin{aligned} T_k &= f(X) \\ X &= T_k \cdot P_k^T + E_k \\ y &= T_k \cdot q_k^T + f_k \\ y &= X \cdot b + f_k \end{aligned} \tag{4.5}$$

4.2 Anwendung der multivariaten Datenanalyse

Dieses Kapitel zeigt die im Rahmen dieser Diplomarbeit umgesetzten Einzelschritte, um mit der Software 'The Unscrambler' eine multivariate Datenanalyse durchzuführen und damit ein Modell zur Qualitätsbeurteilung unbekannter Videosequenzen erstellen zu können.

Mehr Informationen über die lizenzpflichtige Software 'The Unscrambler' finden sich unter [2].

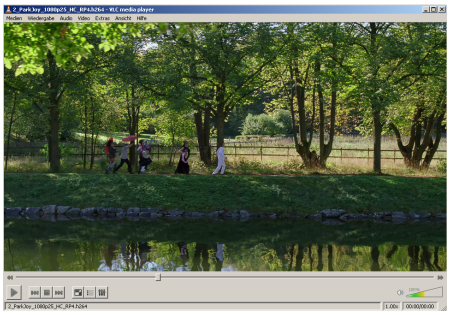
Schritt 1: Vorbereitung (Datenerzeugung aus Testsequenzen)

Zunächst wurden über die modifizierte H.264/AVC Referenzdecoder-Software (siehe Kapitel 3.3) aus 32 Testsequenzen die jeweils 64 Bitstream-Features (siehe Kapitel 3.4) ausgelesen.

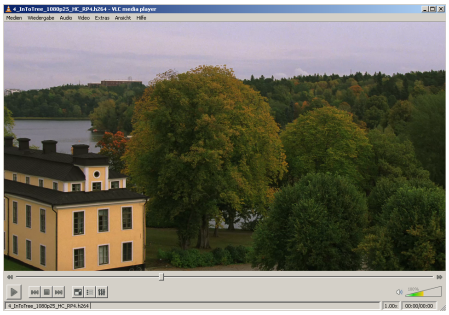
Die 32 Testsequenzen resultierten aus den vier HD-Videosequenzen 'CrowdRun', 'ParkJoy', 'InToTree' und 'OldTownCross' des SVT HD-Test-Sets [21] (siehe auch Abbildung 4.1 auf der nächsten Seite), die mit jeweils acht unterschiedlichen Einstellungen (Profil, Level, Bitrate, Entropiekodierungsmethode, etc.) über die H.264/AVC Referenzencoder-Software [18] H.264-kodiert wurden. Alle 32 Testsequenzen lagen im ungepackten H.264 *byte stream format* vor und hatten die Auflösung 1920 $\times$ 1080 bei 25 Einzelbildern pro Sekunde.

Zudem wurden diese H.264-kodierten Videosequenzen bereits am Lehrstuhl für Datenverarbeitung (LDV) der Technischen Universität München in subjektiven Videotests qualitativ beurteilt, was bekanntlich die Voraussetzung für eine multivariate Datenanalyse ist.

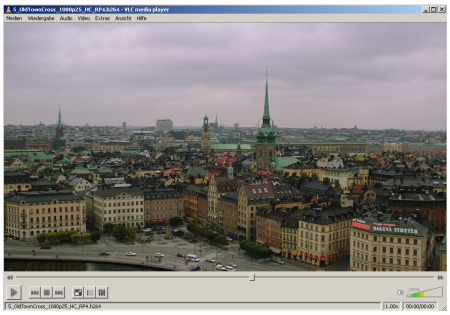
(a) 'CrowdRun'



(b) 'ParkJoy'



(c) 'InToTree'



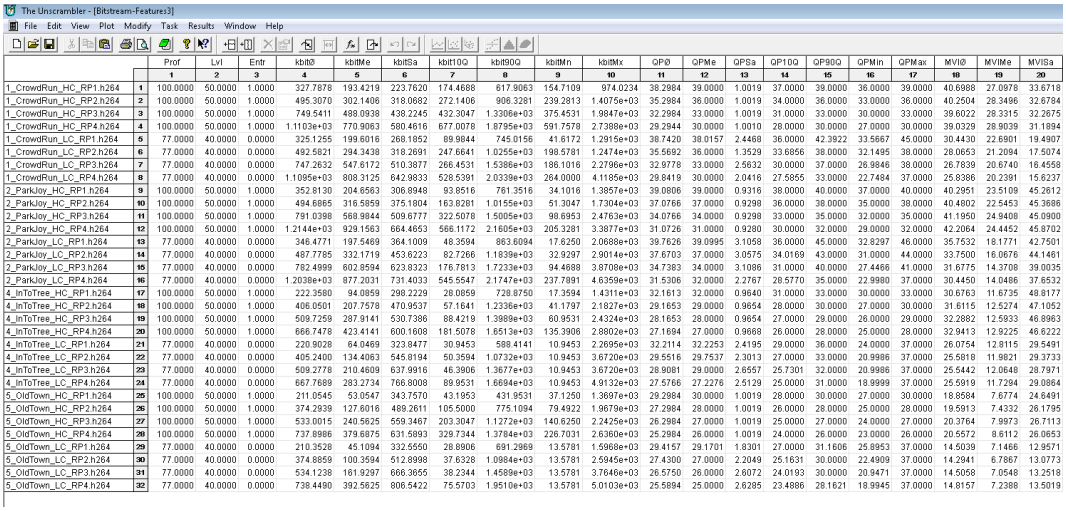
(d) 'OldTownCross'

Abbildung 4.1: Die vier verwendeten HD-Videosequenzen 'CrowdRun', 'ParkJoy', 'InToTree' und 'OldTownCross', welche in jeweils acht unterschiedlichen Qualitätsversionen vorlagen.

Schritt 2: Datenimport

Die in Schritt 1 extrahierten Bitstream-Features der 32 Testsequenzen wurden für den Datenimport in die Software 'The Unscrambler' zusammen mit den Ergebnissen der subjektiven Videotests in eine Microsoft Excel-Tabelle übertragen. Dabei wurde festgestellt, dass die Parameter '$MVMn_{med}$', '$MVMn_{10\text{-}Q}$' und '$MVMn_{90\text{-}Q}$' für alle 32 Testsequenzen den Wert 0 hatten und somit für die multivariate Datenanalyse ignoriert werden konnten. Die damit bereits vorab auf die Größe 32×61 reduzierte Feature-Matrix $\mathbf{X}$ sowie der Qualitätsvektor $\mathbf{y}^{32\times 1}$ wurde anschließend in die Software 'The Unscrambler' importiert.

Abbildung 4.2 auf der nächsten Seite zeigt einen Ausschnitt der Feature-Matrix $\mathbf{X}$.



The Unscrambler - [Bitstream-Features3]

File Edit View Plot Modify Task Results Window Help

	Prof	Lvl	Entr	kbit0	kbitMe	kbitSa	kbit10Q	kbit90Q	kbitMn	kbitMx	QP0	QPMe	QPSa	QP10Q	QP90Q	QPMin	QPMax	MVI0	MVIMe	MVISa
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1_CrowdRun_HC_RP1.h264	100.0000	50.0000	1.0000	327.7878	193.4219	223.7620	174.4688	617.9063	154.7109	974.0234	38.2984	39.0000	1.0019	37.0000	39.0000	36.0000	39.0000	40.6988	27.0978	33.6718
1_CrowdRun_HC_RP2.h264	100.0000	50.0000	1.0000	495.3070	302.1406	318.0682	272.1406	906.3281	239.2813	1.4075e+03	35.2984	36.0000	1.0019	34.0000	36.0000	33.0000	36.0000	40.2504	28.3496	32.6784
1_CrowdRun_HC_RP3.h264	100.0000	50.0000	1.0000	749.5411	488.0938	438.2245	432.3047	1.3306e+03	375.4531	1.9847e+03	32.2984	33.0000	1.0019	31.0000	33.0000	30.0000	33.0000	39.6022	28.3315	32.2675
1_CrowdRun_HC_RP4.h264	100.0000	50.0000	1.0000	1.1103e+03	770.9063	580.4616	677.0078	1.8795e+03	591.7578	2.7388e+03	29.2944	30.0000	1.0010	28.0000	30.0000	27.0000	30.0000	39.0329	28.9039	31.1894
1_CrowdRun_LC_RP1.h264	77.0000	40.0000	0.0000	325.1255	199.6016	268.1852	89.9844	745.0156	41.6172	1.2916e+03	38.7420	38.0157	2.4468	36.0000	42.3922	33.5667	45.0000	30.4430	22.6901	19.4907
1_CrowdRun_LC_RP2.h264	77.0000	40.0000	0.0000	492.5821	294.3438	318.2691	247.6641	1.0255e+03	198.5781	1.2474e+03	35.5692	36.0000	1.3529	33.6856	38.0000	32.1495	38.0000	28.0653	21.2094	17.5074
1_CrowdRun_LC_RP3.h264	77.0000	40.0000	0.0000	747.2632	547.6172	510.3877	266.4531	1.5386e+03	186.1016	2.2796e+03	32.9778	33.0000	2.5632	30.0000	37.0000	26.9846	38.0000	26.7839	20.6740	16.4558
1_CrowdRun_LC_RP4.h264	77.0000	40.0000	0.0000	1.1095e+03	808.3125	642.9833	528.5391	2.0339e+03	264.0000	4.1185e+03	29.8419	30.0000	2.0416	27.5855	33.0000	22.7484	37.0000	25.8386	20.2391	15.6237
2_ParkJoy_HC_RP1.h264	100.0000	50.0000	1.0000	352.8130	204.6563	306.8948	93.8516	761.3516	34.1016	1.3857e+03	39.0806	39.0000	0.9316	38.0000	40.0000	37.0000	40.0000	40.2951	23.5109	45.2612
2_ParkJoy_HC_RP2.h264	100.0000	50.0000	1.0000	494.6865	316.5859	375.1804	163.8281	1.0155e+03	51.3047	1.7304e+03	37.0766	37.0000	0.9298	36.0000	38.0000	35.0000	38.0000	40.4802	22.5453	45.3686
2_ParkJoy_HC_RP3.h264	100.0000	50.0000	1.0000	791.0398	568.9844	509.6777	322.5078	1.5005e+03	98.6953	2.4763e+03	34.0766	34.0000	0.9298	33.0000	35.0000	32.0000	35.0000	41.1950	24.9408	45.0900
2_ParkJoy_HC_RP4.h264	100.0000	50.0000	1.0000	1.2144e+03	929.1563	664.4653	566.1172	2.1605e+03	205.3281	3.3877e+03	31.0726	31.0000	0.9280	30.0000	32.0000	29.0000	32.0000	42.2064	24.4452	45.8702
2_ParkJoy_LC_RP1.h264	77.0000	40.0000	0.0000	346.4771	197.5469	364.1009	48.3594	863.6094	17.8250	2.0688e+03	39.7626	39.0995	3.1058	36.0000	45.0000	32.8297	46.0000	35.7532	18.1771	42.7501
2_ParkJoy_LC_RP2.h264	77.0000	40.0000	0.0000	487.7785	332.1719	453.6223	82.7266	1.1839e+03	32.9297	2.9014e+03	37.6703	37.0000	3.0575	34.0169	43.0000	31.0000	44.0000	33.7500	16.0676	44.1461
2_ParkJoy_LC_RP3.h264	77.0000	40.0000	0.0000	782.4999	602.8594	623.8323	176.7813	1.7233e+03	94.4688	3.8708e+03	34.7383	34.0000	3.1086	31.0000	40.0000	27.4466	41.0000	31.6775	14.3708	39.0035
2_ParkJoy_LC_RP4.h264	77.0000	40.0000	0.0000	1.2038e+03	877.2031	731.4033	545.5547	2.1747e+03	237.7891	4.6359e+03	31.5306	32.0000	2.2767	28.5770	35.0000	22.9980	37.0000	30.4450	14.0486	37.6532
4_InToTree_HC_RP1.h264	100.0000	50.0000	1.0000	222.3580	94.0859	298.2229	28.0859	728.8750	17.3594	1.4311e+03	32.1613	32.0000	0.9640	31.0000	33.0000	30.0000	33.0000	30.6763	11.6735	48.8177
4_InToTree_HC_RP2.h264	100.0000	50.0000	1.0000	406.0501	207.7578	470.9537	57.1641	1.2336e+03	41.1797	2.1827e+03	29.1653	29.0000	0.9654	28.0000	30.0000	27.0000	30.0000	31.6115	12.5274	47.1052
4_InToTree_HC_RP3.h264	100.0000	50.0000	1.0000	509.7259	287.9141	530.7386	88.4219	1.3989e+03	60.9531	2.4324e+03	28.1653	28.0000	0.9654	27.0000	29.0000	26.0000	29.0000	32.2882	12.5933	46.8963
4_InToTree_HC_RP4.h264	100.0000	50.0000	1.0000	666.7478	423.4141	600.1608	181.5078	1.6513e+03	135.3906	2.8802e+03	27.1694	27.0000	0.9668	26.0000	28.0000	25.0000	28.0000	32.9413	12.9225	46.6222
4_InToTree_LC_RP1.h264	77.0000	40.0000	0.0000	220.9028	64.0469	323.8477	30.9453	588.4141	10.9453	2.2695e+03	32.2114	32.2253	2.4195	29.0000	36.0000	24.0000	37.0000	26.0754	12.8115	29.5491
4_InToTree_LC_RP2.h264	77.0000	40.0000	0.0000	405.2400	134.4063	545.8194	50.3594	1.0732e+03	10.9453	3.6720e+03	29.5516	29.7537	2.3013	27.0000	33.0000	20.9986	37.0000	25.5818	11.9821	29.3733
4_InToTree_LC_RP3.h264	77.0000	40.0000	0.0000	509.2778	210.4609	637.9916	46.3906	1.3677e+03	10.9453	3.6720e+03	28.9081	29.0000	2.6557	25.7301	32.0000	20.9986	37.0000	25.5442	12.0648	28.7971
4_InToTree_LC_RP4.h264	77.0000	40.0000	0.0000	667.7689	283.2734	766.8008	89.9531	1.6694e+03	10.9453	4.9132e+03	27.5766	27.2276	2.5129	25.0000	31.0000	18.9999	37.0000	25.5919	11.7294	29.0864
5_OldTown_HC_RP1.h264	100.0000	50.0000	1.0000	211.0545	53.0547	343.7570	43.1953	431.9531	37.1250	1.3697e+03	29.2984	30.0000	1.0019	28.0000	30.0000	27.0000	30.0000	18.8584	7.6774	24.6491
5_OldTown_HC_RP2.h264	100.0000	50.0000	1.0000	374.2939	127.8016	489.2611	105.5000	775.1094	79.4922	1.9679e+03	27.2984	28.0000	1.0019	26.0000	28.0000	25.0000	28.0000	19.5913	7.4332	26.1795
5_OldTown_HC_RP3.h264	100.0000	50.0000	1.0000	533.0015	240.5625	559.3467	203.3047	1.1272e+03	140.6250	2.2425e+03	26.2984	27.0000	1.0019	25.0000	27.0000	24.0000	27.0000	20.3764	7.9973	26.7113
5_OldTown_HC_RP4.h264	100.0000	50.0000	1.0000	737.8986	379.6875	631.5893	329.7344	1.3784e+03	226.7031	2.6360e+03	25.2984	26.0000	1.0019	24.0000	26.0000	23.0000	26.0000	20.5572	8.6112	26.0653
5_OldTown_LC_RP1.h264	77.0000	40.0000	0.0000	210.3528	45.1094	332.5550	28.8906	691.2969	13.5781	1.5968e+03	29.4157	29.1701	1.8301	27.0000	31.1606	25.8953	37.0000	14.5039	7.1466	12.9571
5_OldTown_LC_RP2.h264	77.0000	40.0000	0.0000	374.8859	100.3594	512.8998	37.6328	1.0984e+03	13.5781	2.5945e+03	27.4300	27.0000	2.2049	25.1631	30.0000	22.4909	37.0000	14.2941	6.7867	13.0773
5_OldTown_LC_RP3.h264	77.0000	40.0000	0.0000	534.1238	161.9297	666.3655	38.2344	1.4589e+03	13.5781	3.7646e+03	26.5750	26.0000	2.6072	24.0193	30.0000	20.9471	37.0000	14.5058	7.0548	13.2518
5_OldTown_LC_RP4.h264	77.0000	40.0000	0.0000	738.4490	392.5625	806.5422	75.5703	1.9510e+03	13.5781	5.0103e+03	25.5894	25.0000	2.6285	23.4886	28.1621	18.9945	37.0000	14.8157	7.2388	13.5019

Abbildung 4.2: Ausschnitt der Feature-Matrix **X** im *workspace* der Software 'The Unscrambler'.

Schritt 3: PLSR

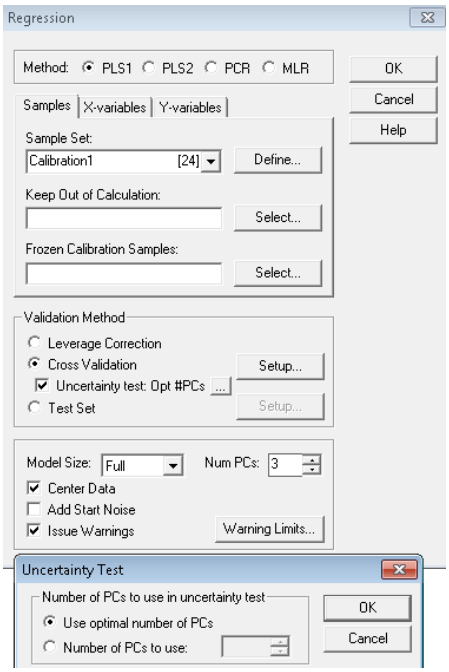
Nach dem Import der Daten folgte der Hauptschritt der multivariaten Datenanalyse, die 'Partial Least Squares Regression' (PLSR).

Um das in dieser Kalibrierungsphase erstellte Modell anschließend verlässlich über das sogenannte Kreuzvalidierungsverfahren (*cross-validation*) validieren zu können, wurden jedoch nicht alle 32 Testsequenzen in die PLSR einbezogen, sondern vorab acht Sequenzen ausgeschlossen und die Regression mit den verbleibenden 24 Sequenzen („Calibration-Set") durchgeführt. Die acht für die Modellvalidierung reservierten Sequenzen („Validation-Set") kamen dabei sinnvollerweise von der gleichen Ursprungssequenz, d.h. sie waren z. B. die acht Variationen der Sequenz 'CrowdRun' (siehe Schritt 1).

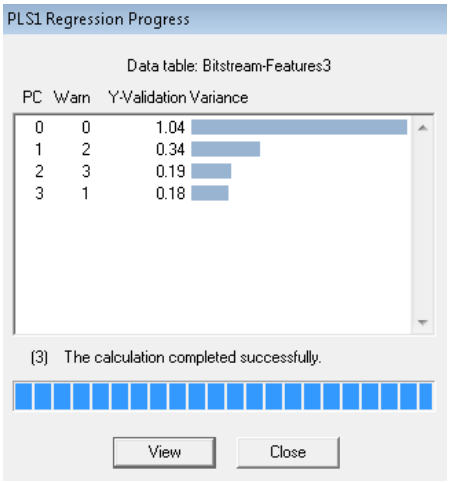
Zur Stabilisation des Modells wurde die PLSR anschließend drei weitere Male durchgeführt, wobei bei jedem Durchlauf andere acht Sequenzen ausgeschlossen wurden und das entsprechend angepasste Calibration-Set verwendet wurde.

Die gewählten Einstellungen bei der Partial Least Squares Regression sind in Abbildung 4.3a auf der nächsten Seite zu sehen. Durch Aktivierung der *'Uncertainty Test'*-Funktion wurde bei der PLSR automatisch über das Kreuzvalidierungsverfahren die optimale Anzahl der Hauptkomponenten (PCs, siehe 4.1.1) bestimmt. Diese erkannte man auch daran, dass die *'Y-Validation Variance'* in Abbildung 4.3b bei der optimalen Anzahl an PCs das erste Minimum erreichte, bzw. ab dieser Anzahl an Hauptkomponenten keine deutliche Verringerung mehr geschah.

Falls bei den vier Durchläufen der PLSR mit den unterschiedlichen Calibration-Sets festgestellt wurde, dass die PC-Anzahl noch nicht optimal gewählt wurde, wurden diese gegebenenfalls mit den optimierten Einstellungen wiederholt.



(a) PLS Regression - Einstellungen



(b) PLSR - Ergebnisse

Abbildung 4.3: Einstellungen und Ergebnisse der PLSR mit der Software 'The Unscrambler'.

Schritt 4: Prädiktion und Modellvalidierung durch 'Cross-Validation'

Im nächsten Schritt wurden die bei der PLSR erstellten Modelle über den Menüpunkt 'Task -> Predict...' in einem Kreuzvalidierungsverfahren validiert. Dabei wurde über jedes der vier Regressions-Modelle die Qualität der Videosequenzen des korrespondierenden Validation-Sets berechnet. Durch Vergleich der berechneten Werte mit den Werten aus den subjektiven Videotests konnte anschließend der Prädiktionsfehler und die Korrelation der 32 Wertepaare bestimmt werden.

Die Kreuzvalidierung (*cross-validation*) stellt eine sehr verlässliche Methode der Validierung dar, da die Validierungsobjekte nicht für die Modellbildung verwendet wurden und damit überprüft werden konnte, ob das Modell für unabhängige Objekte brauchbare Schätzungsergebnisse abliefert.

Schritt 5: Merkmalsreduktion

Abbildung 4.4 auf der nächsten Seite zeigt die Ergebnisübersicht einer Partial Least Squares Regression mit der Software 'The Unscrambler'. In der linken oberen Ecke des Bildaus-

schnittes sieht man die Scores[1], rechts oben findet man die Loadings[2]; im Bereich rechts unten lässt sich die Präzision des Modells über die Korrelationswerte und die angezeigten Regressionsgeraden abschätzen.

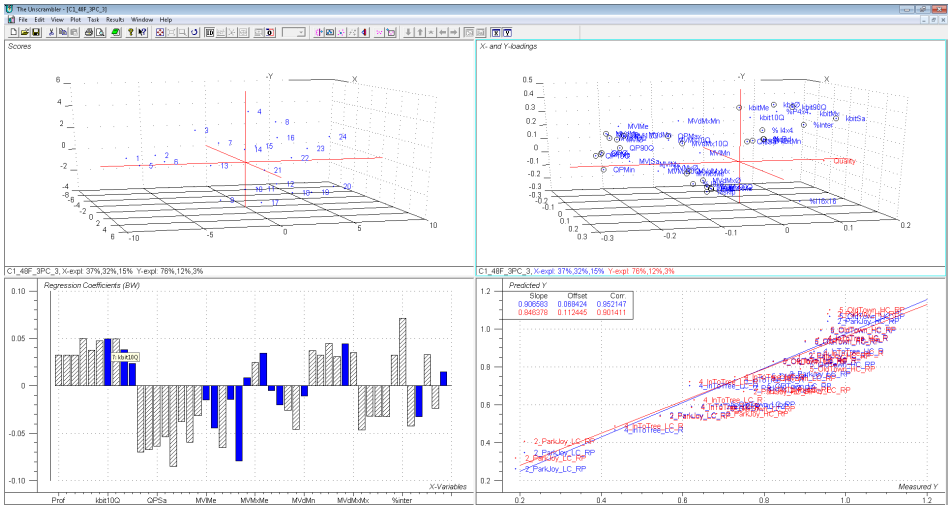


Abbildung 4.4: Ergebnisse einer PLSR mit der Software 'The Unscrambler'.

Da die Modellgenauigkeit bereits durch die Methode in Schritt 4 untersucht wurde, war für diese Arbeit besonders der Abschnitt links unten interessant. Dieser zeigte die Regressionskoeffizienten des Modells, also die Gewichtungen der einzelnen Bitstream-Features in den Hauptkomponenten. Je länger ein Balken, desto stärker floss das entsprechende Merkmal in das Modell ein; negative Gewichtungen bedeuten, dass die Qualität abnimmt, je größer der Wert dieses Parameters ist (dies trifft z. B. für die Quantisierungsparameter-Merkmale zu, siehe Kapitel 2.3.2). Alle Merkmale, die sich bei der Bildung der Hauptkomponenten und damit für das Regressionsmodell als unbrauchbar herausstellten, waren in der Übersicht über die Regressionskoeffizienten blau markiert.

Zur Modelloptimierung wurde daher ausgewertet, welche Bitstream-Features in allen vier Modellen als unbrauchbar gekennzeichnet waren und diese Merkmale aus der Feature-Matrix **X** entfernt.

Schritt 6: Modelloptimierung und Berechnung des finalen Modells

Die Schritte 3-5 wurden anschließend mit der reduzierten Feature-Matrix $\mathbf{X_{red}}$ solange wiederholt, bis keine Bitstream-Features mehr für alle vier Regressionsmodelle als unbrauchbar gekennzeichnet waren. Anschließend wurden die Ergebnisse des 4. Schrittes analysiert und für die Modellbildung der Durchgang verwendet, bei dem die größte Mo-

[1] Beziehung zwischen den Hauptkomponenten und den Testsequenzen, siehe auch Kapitel 4.1.
[2] Beziehung zwischen den Hauptkomponenten und den Bitstream-Features, siehe auch Kapitel 4.1.

dellgenauigkeit[3] festgestellt wurde. Dies musste nicht zwangsläufig der letzte Durchgang sein, was bedeutet, dass das finale Modell damit auch Bitstream-Features enthalten kann, die eigentlich bei allen vier Regressionsmodellen als unbrauchbar markiert waren.

Für den Durchgang mit der größten Modellgenauigkeit wurden die vier Gewichtungsvektoren $\mathbf{b_i}$ ($i \in \{1,2,3,4\}$) der vier Regressionsmodelle kalkuliert und exportiert. Der Gewichtungsvektor $\mathbf{b}$ des finalen Modells wurde schließlich nach Gleichung 4.6 berechnet, indem für jedes Gewicht $b_0 \cdots b_n$ der Mittelwert der entsprechenden Gewichte der vier Gewichtungsvektoren $\mathbf{b_i}$ gebildet wurde.

$$b = \frac{1}{4} \cdot \left(\begin{bmatrix} b_{1,0} \\ b_{1,1} \\ \vdots \\ b_{1,n} \end{bmatrix} + \begin{bmatrix} b_{2,0} \\ b_{2,1} \\ \vdots \\ b_{2,n} \end{bmatrix} + \begin{bmatrix} b_{3,0} \\ b_{3,1} \\ \vdots \\ b_{3,n} \end{bmatrix} + \begin{bmatrix} b_{4,0} \\ b_{4,1} \\ \vdots \\ b_{4,n} \end{bmatrix} \right) \tag{4.6}$$

[3] Als Grundlage für die Modellgenauigkeit wurde die Gesamtkorrelation der 32 Wertepaare verwendet, siehe Schritt 4.

5 Ergebnisse der Qualitätsmetrik

In diesem Kapitel werden die Ergebnisse der multivariaten Datenanalyse präsentiert und ausgewertet. Zunächst wird im nachfolgenden Abschnitt 5.1 eine Übersicht über die im finalen Modell vorhandenen Bitstream-Features und deren Gewichtung gegeben, bevor unter 5.2 die über diese Qualitätsmetrik berechnete Qualitätseinschätzung mit den Ergebnisdaten aus den subjektiven Videotests am Lehrstuhl für Datenverarbeitung der Technischen Universität München verglichen wird. Abschließend werden die Ergebnisse dieser Diplomarbeit unter 5.3 interpretiert und evaluiert.

5.1 Übersicht der finalen Modellparameter

Nachdem die Anwendung der multivariaten Datenanalyse wie im Abschnitt 4.2 beschrieben in insgesamt 11 Durchgängen mit jeweils unterschiedlicher Anzahl an Merkmalen durchgeführt wurde, hat sich bei der Auswertung der Validierungsergebnisse herausgestellt, dass die besten Korrelationswerte mit 48 der ursprünglich 64 Bitstream-Features erreicht wurden. Dies war jedoch nicht der Durchgang mit den wenigsten Merkmalen (das Minimum an verwendeten Bitstream-Features lag bei 39 Merkmalen).

Während bei 50 Bitstream-Features oder mehr die PLSR mit 4 Hauptkomponenten durchgeführt wurde, lag der optimale Wert in den Durchgängen mit weniger als 50 Merkmalen bei 3 PCs. Das finale Modell mit 48 Bitstream-Features basierte demnach auf 3 Hauptkomponenten.

Neben den bereits vor Anwendung der multivariaten Datenanalyse als irrelevant erkannten und somit ausgeschlossenen Bitstream-Features ’$MVMn_{med}$’, ’$MVMn_{10\text{-}Q}$’ und ’$MVMn_{90\text{-}Q}$’ wurden folgende 13 Merkmale für das finale Modell ebenfalls nicht berücksichtigt:

’$MVl_{90\text{-}Q}$’, ’$MVMn_{avg}$’, ’$MVMn_{sd}$’, ’$MVMn_{max}$’, ’$MVMx_{sd}$’, ’$MVMx_{min}$’, ’$MVMx_{max}$’, ’MVd_{sd}’, ’$MVd_{90\text{-}Q}$’, ’MVd_{max}’, ’$MVdMx_{sd}$’ und ’% Intra’, ’% P_8 $\times$ 8’.

Eine Übersicht über die verbliebenen 48 finalen Bitstream-Features sowie deren Gewichtungen liefert Tabelle 5.1 auf der nächsten Seite.

Die genaue Bedeutung der einzelnen Merkmale kann im Kapitel 3.4 nachgelesen werden. Mit Hilfe Gleichung 4.2 konnte somit nach Extraktion dieser 48 Merkmale aus dem Bitstream einer H.264-Videosequenz deren Qualitätsschätzwert berechnet werden.

	Gewichtung		Merkmal	b_{24}	0.000008514766	f_{24}	$MVMx_{avg}$
b_0	1.522121025000			b_{25}	0.000024086755	f_{25}	$MVMx_{med}$
b_1	0.000613825873	f_1	Profile	b_{26}	0.000025454125	f_{26}	$MVMx_{10\text{-}Q}$
b_2	0.001411799900	f_2	Level	b_{27}	-0.000000000658	f_{27}	$MVMx_{90\text{-}Q}$
b_3	0.014117995250	f_3	Entropy	b_{28}	-0.003399421625	f_{28}	MVd_{avg}
b_4	0.000039376511	f_4	$kbit_{avg}$	b_{29}	-0.004190488725	f_{29}	MVd_{med}
b_5	0.000037409327	f_5	$kbit_{med}$	b_{30}	-0.008260916025	f_{30}	$MVd_{10\text{-}Q}$
b_6	0.000061723944	f_6	$kbit_{sd}$	b_{31}	-0.004817729750	f_{31}	MVd_{min}
b_7	0.000059136347	f_7	$kbit_{10\text{-}Q}$	b_{32}	0.000034156459	f_{32}	$MVdMx_{avg}$
b_8	0.000022219557	f_8	$kbit_{90\text{-}Q}$	b_{33}	0.000031738951	f_{33}	$MVdMx_{med}$
b_9	0.000062160026	f_9	$kbit_{min}$	b_{34}	0.000031455675	f_{34}	$MVdMx_{10\text{-}Q}$
b_{10}	0.000003980148	f_{10}	$kbit_{max}$	b_{35}	0.000030044604	f_{35}	$MVdMx_{90\text{-}Q}$
b_{11}	-0.003864684375	f_{11}	QP_{avg}	b_{36}	0.000028293649	f_{36}	$MVdMx_{min}$
b_{12}	-0.003603678250	f_{12}	QP_{med}	b_{37}	0.000026565068	f_{37}	$MVdMx_{max}$
b_{13}	-0.018989462500	f_{13}	QP_{sd}	b_{38}	-0.042094641750	f_{38}	qpd_{avg}
b_{14}	-0.003042883300	f_{14}	$QP_{10\text{-}Q}$	b_{39}	-0.000561445370	f_{39}	% qpd
b_{15}	-0.004115063400	f_{15}	$QP_{90\text{-}Q}$	b_{40}	-0.042080475750	f_{40}	% I-Slices
b_{16}	-0.001843248400	f_{16}	QP_{min}	b_{41}	-0.001429322725	f_{41}	% P-Slices
b_{17}	-0.003354319900	f_{17}	QP_{max}	b_{42}	0.001382369450	f_{42}	% B-Slices
b_{18}	-0.000991776785	f_{18}	MVl_{avg}	b_{43}	0.002366921150	f_{43}	% Inter
b_{19}	-0.000570992720	f_{19}	MVl_{med}	b_{44}	-0.001207029303	f_{44}	% Skip
b_{20}	-0.000649316588	f_{20}	MVl_{sd}	b_{45}	-0.000844287618	f_{45}	% I_16 $\times$ 16
b_{21}	-0.002994685550	f_{21}	$MVl_{10\text{-}Q}$	b_{46}	0.000230966340	f_{46}	% I_8 $\times$ 8
b_{22}	-0.001084648548	f_{22}	MVl_{min}	b_{47}	-0.000148648913	f_{47}	% I_4 $\times$ 4
b_{23}	-0.000211019785	f_{23}	MVl_{max}	b_{48}	0.001417432095	f_{48}	% P_4 $\times$ 4

Tabelle 5.1: Übersicht der Modellparameter

5.2 Validierung der Qualitätsmetrik

Tabelle 5.2 auf der nächsten Seite vergleicht die Referenz-Qualitätswerte mit den über das Modell berechneten Prädiktions-Qualitätswerten.

Die Referenzwerte wurden in subjektiven Videotests am Lehrstuhl für Datenverarbeitung der Technischen Universität München ermittelt und beschreiben den Mittelwert der Beurteilung von 18 Testpersonen, die die Qualität der 32 H.264-Videosequenzen auf einer Skala von 0 bis 1 bewerteten. Die letzte Spalte der Tabelle 5.2 veranschaulicht den Prädiktionsfehler, also die prozentuale Abweichung des Prädiktionswert vom Referenzwert. Ist der Prädiktionsfehler negativ, wurde der Qualitätswert unterschätzt; positive Werte bedeuten eine Überschätzung.

Der durchschnittliche Prädiktionsfehler betrug 9,67%; der Korrelationskoeffizient lag bei 0,957.

Sequenz	Referenz	Prädiktion	Abweichung
CrowdRun_1080p25_HC_RP1.h264	0.506	0.541	+7.02%
CrowdRun_1080p25_HC_RP2.h264	0.694	0.660	-4.92%
CrowdRun_1080p25_HC_RP3.h264	0.828	0.810	-2.10%
CrowdRun_1080p25_HC_RP4.h264	0.933	0.980	+4.96%
CrowdRun_1080p25_LC_RP1.h264	0.261	0.303	+16.21%
CrowdRun_1080p25_LC_RP2.h264	0.556	0.482	-13.20%
CrowdRun_1080p25_LC_RP3.h264	0.617	0.590	-4.38%
CrowdRun_1080p25_LC_RP4.h264	0.783	0.763	-2.64%
ParkJoy_1080p25_HC_RP1.h264	0.683	0.581	-14.98%
ParkJoy_1080p25_HC_RP2.h264	0.750	0.679	-9.53%
ParkJoy_1080p25_HC_RP3.h264	0.886	0.834	-5.89%
ParkJoy_1080p25_HC_RP4.h264	0.956	1.006	+5.24%
ParkJoy_1080p25_LC_RP1.h264	0.189	0.267	+41.50%
ParkJoy_1080p25_LC_RP2.h264	0.211	0.350	+65.57%
ParkJoy_1080p25_LC_RP3.h264	0.544	0.539	-0.93%
ParkJoy_1080p25_LC_RP4.h264	0.839	0.750	-10.62%
InToTree_1080p25_HC_RP1.h264	0.728	0.732	+0.55%
InToTree_1080p25_HC_RP2.h264	0.856	0.845	-1.29%
InToTree_1080p25_HC_RP3.h264	0.922	0.887	-3.84%
InToTree_1080p25_HC_RP4.h264	0.933	0.950	+1.74%
InToTree_1080p25_LC_RP1.h264	0.433	0.491	+13.40%
InToTree_1080p25_LC_RP2.h264	0.622	0.618	-0.63%
InToTree_1080p25_LC_RP3.h264	0.628	0.651	+3.70%
InToTree_1080p25_LC_RP4.h264	0.617	0.725	+17.53%
OldTownCross_1080p25_HC_RP1.h264	0.889	0.822	-7.52%
OldTownCross_1080p25_HC_RP2.h264	0.900	0.931	+3.45%
OldTownCross_1080p25_HC_RP3.h264	0.939	0.989	+5.29%
OldTownCross_1080p25_HC_RP4.h264	0.961	1.063	+10.63%
OldTownCross_1080p25_LC_RP1.h264	0.694	0.607	-12.57%
OldTownCross_1080p25_LC_RP2.h264	0.783	0.700	-10.67%
OldTownCross_1080p25_LC_RP3.h264	0.794	0.751	-5.48%
OldTownCross_1080p25_LC_RP4.h264	0.822	0.833	+1.36%

Tabelle 5.2: Vergleich der in subjektiven Videotests ermittelten Qualitätswerte (Referenz) mit den Schätzwerten des Modells (Prädiktion). Die letzte Spalte gibt den Prädiktionsfehler an.

Zur besseren Veranschaulichung ist in Abbildung 5.1 ein Punktdiagramm zu sehen, das die geschätzte Qualität (y-Wert) über der Referenzqualität (x-Wert) für alle 32 Videosequenzen anzeigt. Entspricht der Schätzwert exakt dem Referenzwert, so liegt der entsprechende Punkt im Diagramm auf der eingezeichneten Gerade $y = x$. Liegt der Punkt oberhalb der Gerade, wird die Qualität überschätzt; liegt er unterhalb der Gerade, dann wird die Qualität unterschätzt.

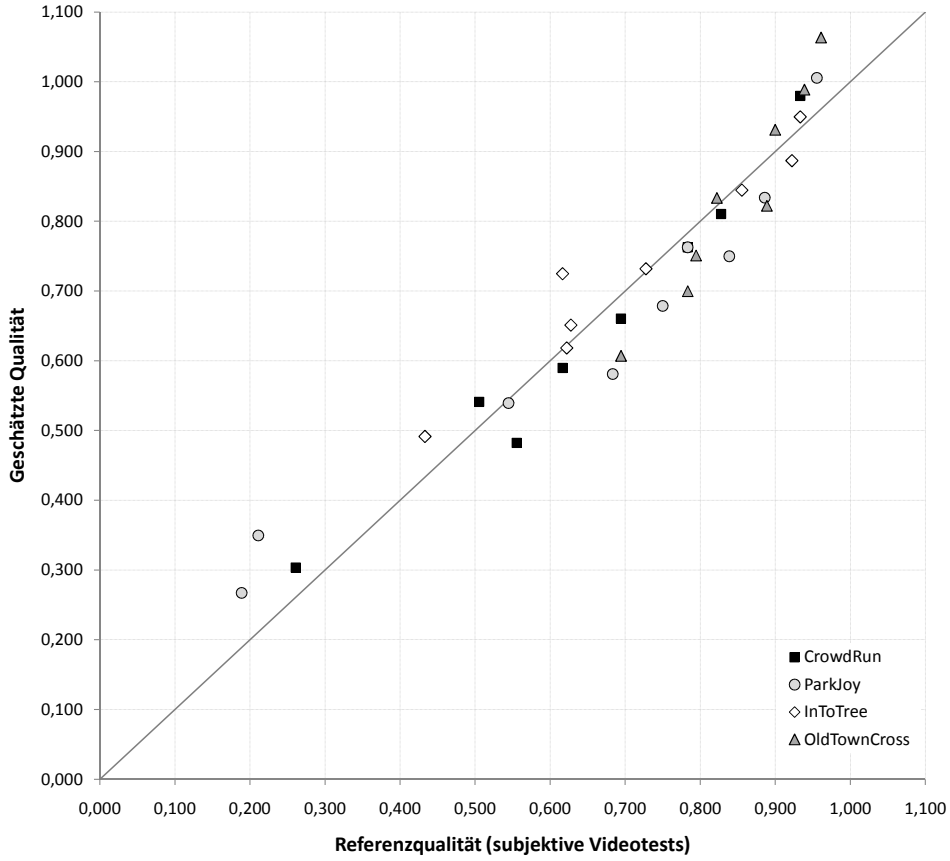


Abbildung 5.1: Graphischer Vergleich der Referenz-Qualitätswerte (x) mit den Prädiktions-Qualitätswerten (y).

5.3 Evaluierung der Ergebnisse

Wie anhand der vorgestellten Ergebnisse der Modellvalidierung (siehe Tabelle 5.2 und Diagramm 5.1) ersichtlich ist, liefert die in dieser Arbeit entworfene Qualitätsmetrik sehr gute Ergebnisse.

Die Korrelation der Qualitätsschätzung mit den Referenzwerten der subjektiven Videotests war überdurchschnittlich hoch; der Korrelationskoeffizient betrug 0,957 und übertrifft damit sogar knapp den Wert der Qualitätsmetrik von A. Rossholm et al. , welche einen Korrelationskoeffizienten von 0,95 erreichten [16]. Allerdings sollte dabei auch berücksichtigt werden, dass in das finale Modell, welches die Basis dieser Auswertung war, durch die Mittelwertbildung der vier Teilmodelle wieder alle 32 Videosequenzen einflossen.

Der „saubere“[1] Gesamtkorrelationskoeffizient, der aus den Prädiktionsdaten der Validierungsphase (siehe 4.2 - Schritt 4) für den Durchgang der größten Modellgenauigkeit berechnet wurde, betrug 0,913 und ist für *no-reference*-Verfahren immer noch sehr gut.

Betrachtet man Tabelle 5.2, lässt sich feststellen, dass über das finale Regressionsmodell die Qualität der verfügbaren Videosequenzen bis auf zwei Ausreißer sehr genau geschätzt werden konnte. Die beiden Ausreißer waren dabei die Sequenzen mit der geringsten Qualität; beide waren aus der Ursprungssequenz 'ParkJoy'. Dass die prozentuale Abweichung für diese beiden Videosequenzen derart hoch war, lässt sich einerseits dadurch erklären, dass bei gleicher relativer Abweichung (Differenz der beiden Werte) die prozentuale Abweichung automatisch größer wird, wenn der Absolutwert der Qualität gering ist. Andererseits erkennt man in Abbildung 5.1, dass bis auf drei Videosequenzen, unter denen auch die beiden Ausreißer waren, alle anderen 29 Sequenzen deutlich höhere Qualität aufwiesen, und das Regressionsmodell damit vermutlich für diese 29 Sequenzen optimiert wurde.

Trotz der beiden Ausreißer war der Mittelwert des Prädiktionsfehlers dennoch sehr gering; im Schnitt wurde die Qualität der 32 Videosequenzen über die Qualitätsmetrik mit weniger als 10 Prozent Abweichung vorausgesagt. Dabei wurden Sequenzen mit niedriger oder sehr hoher Qualität vom Modell meist überschätzt, während Videosequenzen mittlerer Qualität eher unterschätzt wurden.

Von den ursprünglich 64 extrahierten Bitstream-Features wurden während der Kalibrierungsphase 16 Merkmale aus dem Modell ausgeschlossen. Darunter befanden sich alle sechs Variationen des Merkmals 'MVMn', welches den minimalen Bewegungsvektor pro interkodiertem Slice erfasste. Bei genauer Betrachtung ist dies durchaus nachvollziehbar, da dieser Minimalwert in den meisten Fällen gegen Null strebt, und daher keine große Variation und Aussagekraft in diesen Daten lag. Dass die Merkmale '% Intra' und '% P_8 $\times$ 8' als unbrauchbar erkannt wurden, hat keine große Bedeutung, da beide linear abhängig von verwendeten Merkmalen sind ('% Inter' und '% Skip' bzw. '% P_4 $\times$ 4'). Die restlichen acht ausgeschlossenen Merkmale waren ebenfalls durch mindestens ein anderes Statistikmaß ausgedrückt im finalen Modell vorhanden.

In diesem Modell waren in der Software 'The Unscrambler' zwar immer noch sechs Merkmale ('$kbit_{max}$', 'MVl_{sd}', 'MVl_{max}', '$MVMx_{10\text{-}Q}$', '$MVdMx_{min}$' und '% I_16 $\times$ 16') bei allen vier Teilmodellen als unbrauchbar markiert, allerdings führte deren Entfernung zu einem instabileren Modell und schlechteren Ergebnissen.

[1] Mit „sauber“ ist gemeint, dass der Wert aus Daten resultiert, die durch das Kreuzvalidierungsverfahren ermittelt wurden.

Wirft man einen Blick auf die Gewichtungen der 48 finalen Bitstream-Features in Tabelle 5.1 und multipliziert jedes Gewicht mit dem Mittelwert der 32 Werte, die für das entsprechende Merkmal von den 32 Testsequenzen ausgelesen wurden, lässt sich über das Ergebnis eine Aussage über die Relevanz des entsprechenden Bitstream-Features zur Qualitätsschätzung treffen. Demnach sind folgende 10 Bitstream-Features am aussagekräftigsten in Bezug auf die Bildqualität (in Klammern steht jeweils das Ergebnis der Multiplikation des Gewichts mit dem Mittelwert der 32 Merkmalswerte):

1. % I-Slices (-0,346)
2. $QP_{90\text{-}Q}$ (-0,139)
3. QP_{avg} (-0,122)
4. QP_{max} (-0,119)
5. QP_{med} (-0,114)
6. % B-Slices (0,099)
7. % Inter (0,091)
8. $QP_{10\text{-}Q}$ (-0,090)
9. Level (0,064)
10. % Skip (-0,055)

An dieser Liste lässt sich erkennen, dass insbesondere der Quantisierungsparameter eine sehr große Relevanz für die Qualität einer Videosequenz aufweist, aber auch die Anteile der verschiedenen Slice- und Makroblock-Arten großen Einfluss auf die Bildqualität haben. Die Bitrate ('kbit') hingegen, die man intuitiv sehr weit oben in dieser Liste erwartet hätte, ist erst im oberen Mittelfeld der nach Relevanz sortierten Liste zu finden.

Obwohl bei der H.264-Kodierung der 32 Testsequenzen darauf Wert gelegt wurde, alle acht Videosequenzen einer Ursprungssequenz mit unterschiedlichen Einstellungen zu kodieren und damit eine größtmögliche Variation in den Merkmalen der Testsequenzen zu erzeugen, wiesen die Bitstream-Features der Testsequenzen dennoch teilweise begrenzte Wertebereiche auf, die die Anwendbarkeit der Qualitätsmetrik einschränken. So war das Encoding-Profil entweder 'Main' (mit Level 4.0) oder 'High' (mit Level 5.0) und die Bitrate umgerechnet im Bereich zwischen 5 Mbit/s und 30 Mbit/s.

Bei Verwendung des Modells ist deshalb darauf zu achten, dass die Videosequenz, auf die die Qualitätsmetrik angewandt wird, ähnliche Eigenschaften aufweist, um wirklich aussagekräftige Resultate zu erhalten.

6 Zusammenfassung und Ausblick

In dieser Diplomarbeit wurde ein effizientes *no-reference*-Verfahren zur Qualitätsbeurteilung von H.264-kodierten HD-Videosequenzen vorgestellt, welches völlig ohne Informationen aus dem dekodierten Bildmaterial auskommt. Dazu wurden über die entsprechend modifizierte H.264/AVC Referenzdecoder-Implementierung aus dem H.264-Bitstream von 32 HD-Testsequenzen insgesamt 64 charakteristische Merkmale ausgelesen. Diese Bitstream-Features wurden zusammen mit den subjektiven Qualitätsbeurteilungen der HD-Testsequenzen, welche vorab in Videotests am Lehrstuhl für Datenverarbeitung der Technischen Universität München ermittelt wurden, verwendet, um mithilfe einer multivariaten Datenanalyse ein 'Partial Least Squares' (PLS) Regressionsmodell zu finden, das den Zusammenhang zwischen den Bitstream-Features und der Qualität der entsprechenden Videosequenz beschreibt. Bei der Kalibrierungsphase des Modells stellten sich 48 der 64 Bitstream-Features als brauchbar für die Qualitätsmetrik heraus und fanden daher für das finale Regressionsmodell Verwendung.

Besonders stark gewichtet sind in diesem Modell, das auf 3 Hauptkomponenten basiert, die Merkmale '% I-Slices', '$QP_{90\text{-}Q}$', 'QP_{avg}', 'QP_{max}', 'QP_{med}' und '% B-Slices'; die Bitrate der Videosequenzen sowie die Bewegungsvektoren hingegen spielen nur eine untergeordnete Rolle.

Die Modellvalidierung ergab einen Korrelationskoeffizient von 0,957 zwischen den Referenzwerten und den geschätzten Werten, womit die Qualitätsmetrik sehr gute Ergebnisse erzielte. Im Durchschnitt wurde die Qualität der überprüften Videosequenzen durch das Regressionsmodell auf $\pm 9,67\%$ genau abgeschätzt.

Verbesserungsbedarf besteht jedoch noch in Hinblick auf die Echtzeitfähigkeit des *feature-extractors*. Nachdem diese Implementierung auf Basis der relativ unperformanten H.264/AVC Referenz-Software entstand und für Code-Optimierung im Rahmen dieser Diplomarbeit nur begrenzt Zeit war, muss hier auf jeden Fall noch nachgebessert werden, falls die Software in einer Echtzeitumgebung eingesetzt werden soll.

Ein Durchlauf zum Verarbeiten und Auslesen der Informationen aus einem Einzelbild der HD-Videosequenzen (Auflösung: 1920×1080) dauerte im Schnitt circa 300 ms, was einer Geschwindigkeit von circa 3 fps (*frames per second*) entsprach.

Eine weitere Einschränkung der Qualitätsmetrik war, dass für die PLSR jedes Bitstream-Feature genau einem Parameter (Gleitkommawert) entsprechen musste. Um dies zu erreichen, konnte z. B. nicht jeder Bewegungsvektor einzeln betrachtet werden, sondern es musste von allen Bewegungsvektoren eines Slices jeweils der Mittelwert gebildet werden, die Mittelwerte mussten dann wiederrum auf Slice-Ebene durch verschiedene Maße der Statistik zu einem Wert zusammengefasst werden mussten.

Denkbar wäre daher für Arbeiten in der Zukunft eine Erweiterung der Qualitätsmetrik auf die Makroblock-Ebene, was bedeuten würde, dass jedes Bitstream-Feature durch eine Liste *(array)* repräsentiert wird, und somit eine 3D-Modellierung nötig wäre.

Literaturverzeichnis

1. I. BT. 500-11, Methodology for the Subjective Assessment of the Quality of Television Pictures, Recommendation ITU-R BT. 500-11, ITU Telecom. In *Sector of ITU*, 2002.

2. Camo. The Unscrambler®: Comprehensive Multivariate Analysis and Experimental Software. URL `http://www.camo.com/rt/Products/Unscrambler/unscrambler.html`.

3. I. DIS. 10918-1 Digital compression and coding of continuous-tone still images (JPEG). In *CCITT Recommendation T*, 81, 1992.

4. A. Eden. No-reference estimation of the coding PSNR for H. 264-coded sequences. In *Consumer Electronics, IEEE Transactions on*, 53(2), S. 667–674, 2007.

5. B. Flannery, W. Press, S. Teukolsky und W. Vetterling. Numerical recipes in C. In *Press Syndicate of the University of Cambridge, New York*, 1992.

6. A. Heyna, M. Briede und U. Schmidt. *Datenformate im Medienbereich*. Hanser Verlag, 2003.

7. Q. Huynh-Thu und M. Ghanbari. Scope of validity of PSNR in image/video quality assessment. In *Electronics Letters*, 44, S. 800, 2008.

8. ITU-T. Recommendation and final draft international standard of joint video specification (ITU-T Rec. H. 264| ISO/IEC 14496-10 AVC). In *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVTG050*, 2010.

9. J. Jain und A. Jain. Displacement measurement and its application in interframe image coding. In *IEEE Transactions on communications*, 29(12), S. 1799–1808, 1981.

10. C. Keimel, T. Oelbaum und K. Diepold. No-reference video quality evaluation for high-definition video. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, S. 1145–1148. IEEE, 2009.

11. H. Martens und M. Martens. *Multivariate analysis of quality: an introduction*. John Wiley & Sons Inc, 2001. ISBN 0471974285.

12. J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer und T. Wedi. Video coding with H. 264/AVC: tools, performance, and complexity. In *Circuits and Systems magazine, IEEE*, 4(1), S. 7–28, 2005.

13. S. Péchard, D. Barba und P. Le Callet. In Video Quality Model based on a spatiotemporal features extraction for H. 264-coded HDTV sequences, 2007.

14. C. Poynton. *Digital video and HDTV: algorithms and interfaces*. Morgan Kaufmann, 2003.

15. I. Rec. H. 264| ISO/IEC 14496-10 AVC. In *Advanced video coding for generic audiovisual services*, 2005.

16. A. Rossholm und B. Lövström. A new video quality predictor based on decoder parameter extraction. In *International Conference on Signal Processing and Multimedia Applications*. Inst Syst & Technologies Informat, Control & Commun, 2008.

17. M. Slanina, V. Ricny und R. Forchheimer. A Novel Metric for H. 264/AVC No-Reference Quality Assessment. In *Systems, Signals and Image Processing, 2007 and 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services. 14th International Workshop on*, S. 114–117. IEEE, 2007.

18. K. Suehring. H.264/AVC JM Reference Software. URL `http://iphome.hhi.de/suehring/tml/`.

19. O. Sugimoto, S. Naito, S. Sakazawa und A. Koike. Objective perceptual video quality measurement method based on hybrid no reference framework. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, S. 2237–2240. IEEE, 2010.

20. G. Sullivan und T. Wiegnad. Video compression-From concepts to the H. 264/AVC standard. In *Proceedings of the IEEE*, 93(1), S. 18–31, 2005.

21. SVT. The SVT High Definition Multi Format Test Set. URL `ftp://vqeg.its.bldrdoc.gov/HDTV/SVT_MultiFormat/`.

22. T. Wiegand, G. Sullivan, G. Bjontegaard und A. Luthra. Overview of the H. 264/AVC video coding standard. In *IEEE Transactions on circuits and systems for video technology*, 13(7), S. 560–576, 2003.

23. Wikipedia. Bildergruppe. . URL `http://de.wikipedia.org/wiki/Bildergruppe`.

24. Wikipedia. Farbunterabtastung. . URL `http://de.wikipedia.org/wiki/Farbunterabtastung`.